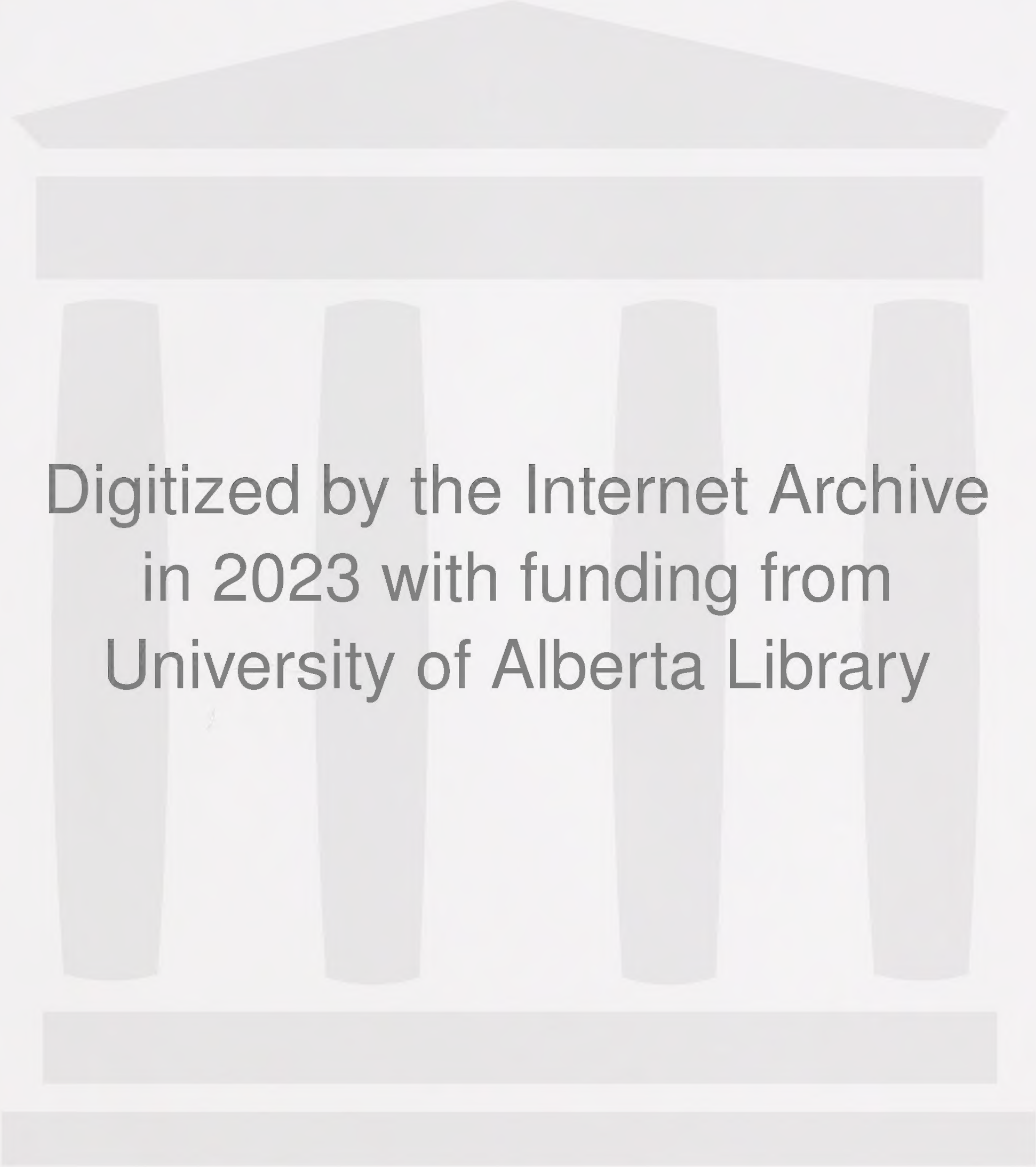

QUAECUMQUE VERA



THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR EWE LEE CHOOT

TITLE OF THESIS SELF-ORGANIZING LINEAR SEARCH HEURISTICS FOR A DYNAMIC FIXED-SIZE LIST

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED FALL 1983

THE UNIVERSITY OF ALBERTA

SELF-ORGANIZING LINEAR SEARCH HEURISTICS FOR A DYNAMIC FIXED-SIZE LIST

by

© EWE LEE CHOOT

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL 1983

THE UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled SELF-ORGANIZING LINEAR SEARCH HEURISTICS FOR A DYNAMIC FIXED-SIZE LIST submitted by EWE LEE CHOOT in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE in COMPUTING SCIENCE.

Abstract

Self-organizing heuristics are often adapted to maintain a sequential list in near optimal order when the frequency of accessing each element is not known in advance. An account of the existing schemes is given, and the self-organizing heuristics are modified to handle the case of a dynamic fixed-size list, where only a subset of elements is maintained in the sequential list at a given time. An application of this modified scheme is the construction of paging algorithms for managing storage hierarchies.

This thesis examines the performance of the modified scheme, with respect to the average time required to search for an element. The analysis of the performance involves modelling the stream of requests as a Markov chain. The class of modified move to front heuristics is studied in detail. Empirical evidence, including numerical examples and simulation results, suggests that the class of modified transposition heuristics is asymptotically more efficient than the corresponding move to front heuristics.

Acknowledgements

I would like to thank my supervisor, Dr. Francis Chin, for his guidance and constructive criticisms throughout all stages of this work. I am also grateful to the comments provided by the examination committee member, Dr. L.J. White, Dr. J.R. Sampson, and Professor M. Hlynka.

I would also like to express my thanks to all those who provided me with encouragement, and especially to Ming and all members of my family, for their continual support.

Table of Contents

Chapter Page

List of Tables

Table Page

I. Introduction

Suppose we have a set of n records, $R_1, R_2, \ldots, R_n$, which is arranged in some arbitrary serial order π, where R_i is in position $\pi(i)$, for $1 \le i \le n$. At each time instant, $t_1, t_2, \ldots$, a record is asked for. To fetch the requested record R_i, the linear list is searched sequentially, starting from position 1, until the record is found in position $\pi(i)$.

Assume that each record R_i has a probability p_i of being selected, and that p_i satisfies the following conditions

1. $p_i > 0$ for all i
2. $\sum_{i=1}^{n} p_i = 1$
3. p_i remains unchanged and independent at all times, i.e., successive requests are probabilistically independent of each other and of the ordering of the records (INDEPENDENCE ASSUMPTION).

In terms of the expected number of searches required to retrieve a record, a random arrangement would be expected to perform poorly, as it requires $(n+1)/2$ searches on the average. Unless all the records have an equal chance of being accessed, a rearrangement of the records could reduce the expected search cost. The obvious approach would be to order the records by non-increasing retrieval probabilities. However, in practice, these probabilities are rarely known a priori, and optimal ordering of the records cannot be done in advance. Under this circumstance, heuristics that dynamically rearrange the records are often used to reduce

expected search cost. The idea behind these heuristics is to utilize previous accesses as a basis for rearranging the records into a less costly ordering.

One approach would be to count the number of requests for a record, and rearrange the records in non-increasing order by request counts. Given enough time, by the law of large numbers, records with higher retrieval probabilities will have higher access frequencies, and hence optimal ordering could be reached. However, the major problem with this approach is the extra amount of storage incurred to keep the counts. The cost of maintaining the counts is often prohibitive (see Knuth[10]), and one has to resort to memory free self-organizing heuristics.

A. Memory Free Self-Organizing Heuristics

The two most frequently mentioned memory free self-organizing heuristics are the *move to front* and the *transposition* schemes. These two heuristics have been studied by Bitner[1], Burville and Kingman[2], Gonnet, Munro and Suwanda[5], Hendricks[6,7,8], Knuth[10], Lam, Leung, and Siu[11], McCabe[12], Rivest[13], and Tanenbaum[14]. The basic approach of these two heuristics is to perform a permutation upon the existing order of the records, every time a record is requested, according to some specific rules which will be explained later.

Define a self-organizing scheme to be a set of permutations, $\{\tau_i, 1 \leq i \leq n\}$, where τ_i is the permutation performed when record in position i is requested. After application of τ_i, the record in position j will be moved to position $\tau_i(j)$. The two schemata are given below.

Move to Front Heuristics (MTF)

$$\tau_i(j) = \begin{cases} j + 1, & 1 \leq j \leq i-1 \\ 1, & j = i \\ j, & j > i \end{cases}$$

i.e., when a record in position i is accessed, this requested record will be placed in position 1, while records in position 1 through i-1 are moved back one position to make room for it. The ordering remains the same if the requested record heads the list.

Transposition Heuristics (TR)

$$\tau_i(j) = \begin{cases} j - 1, & \text{if } j = i \text{ and } j \neq 1 \\ j + 1, & \text{if } j = i-1 \\ j, & \text{otherwise} \end{cases}$$

i.e., a requested record is exchanged with the one preceding it. Similarly here, requesting a record in position 1 leaves the ordering unchanged.

McCabe[11] has shown that asymptotically the expected search time, μ, for the MTF scheme satisfies

$$\mu(\text{MTF}) = 1 + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{p_i p_j}{p_i + p_j}$$

Burville and Kingman[2] have shown that, if $p_1 \geq p_2 \geq p_3 \geq \ldots \geq p_n$ then

$$m \leq \mu(\text{MTF}) < 2m-1, \qquad m = \sum_{i=1}^{n} i p_i$$

where m is the expected search cost for the optimal ordering $R_1R_2R_3...R_n$. The above inequality shows that MTF never does more than twice the work done with the optimal ordering.

The MTF scheme does exhibit considerable savings over random arrangement. As an example, Knuth[10] showed that, if the retrieval probabilities obey Zipf's distribution, i.e. $p_i = 1/(iH_n)$, where H_n is the nth harmonic number ($H_n = \Sigma_{i=1}^{n} 1/i$), $\mu(MTF) \cong 2n/\log_2 n$. This cost is substantially better than $(n+1)/2$, and is about $\ln 4 \cong 1.386$ times as many comparisons as would be obtained in the optimal arrangement.

The expected search cost for the transposition heuristic can be evaluated for a specific retrieval distribution (see Hendricks[8]). However, no concise general expression is given. Rivest[13] has shown that the expected search cost, $\mu(TR)$, is always more efficient than $\mu(MTF)$, except when $n=2$ or all p_i's are equal. Rivest also conjectures that TR is asymptotically more efficient than any other memory-free self-organizing heuristics, for any retrieval probability distribution. Various evidence to support this conjecture is presented; however, no direct proof has been given.

B. Self-Organizing Heuristics with Limited Storage

Gonnet, Munro, and Suwanda[5] later showed that, with the use of a very small amount of storage to remember a few previous requests, expected search cost could be further

reduced. The notion of *k in a row* heuristic is introduced. The first approach is to apply the transposition (move-to-front or any other) heuristic only if the same element is accessed k times in a row. This approach would use $\log_2(nk)$ bits of extra storage instead of the $O(n)$ or more bits required for the counter scheme. This is known as the *simple k* heuristic. A slight variation of this scheme, known as the *batched k* heuristic, will be explained later.

Simple k Heuristics

The expected search cost of the simple k MTF heuristic is given in [5]

$$\mu_k(\text{MTF}) = 1 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{p_j{}^k p_i \sum_{t=0}^{k-1} p_i{}^t + p_i{}^k p_j \sum_{t=0}^{k-1} p_j{}^t}{p_j{}^k \sum_{t=0}^{k-1} p_i{}^t + p_i{}^k \sum_{t=0}^{k-1} p_j{}^t}$$

It is also shown that $\mu_{k+1}(\text{MTF}) \leq \mu_k(\text{MTF})$, for $k \geq 1$. The following example, taken directly from the paper, demonstrates the performance of the scheme

$\mu_2(\text{MTF}) \leq 1.36602$ Opt

$\mu_3(\text{MTF}) \leq 1.27388$ Opt

$\mu_4(\text{MTF}) \leq 1.22788$ Opt

Although the exact search cost for the simple k heuristic with transposition rule, $\mu_k(\text{TR})$, has not been obtained, the cost has been shown to be better than the simple k move to front heuristics. The expected search cost, $\mu_k(\text{TR})$, is also shown to decrease as k increases.

$$\mu_k(\mathrm{TR}) \le \mu_k(\mathrm{MTF}) \text{ and } \mu_{k+1}(\mathrm{TR}) \le \mu_k(\mathrm{TR})$$
for all distributions, $k \ge 1$.

Batched k Heuristics

This is a slight variation of the simple k heuristic in that requests are batched into groups of k elements, and reorganizing occurs only when the k requests in a batch are for the same element. In other words, when an element is accessed k times in a row, the transposition (or move to front) rule may or may not be applied, depending on whether the k accesses belong to the same batch.

In [5], the expected search cost for the MTF scheme under the batched k heuristic is given as

$$\mu'_k(\mathrm{MTF}) = 1 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{p_j{}^k p_i + p_i{}^k p_j}{p_j{}^k + p_i{}^k}$$

Similarly here, it is shown that $\mu'_{k+1}(\mathrm{MTF}) \le \mu'_k(\mathrm{MTF})$, for $k \ge 1$. In addition, the batched k heuristic is shown to perform better than the corresponding simple k heuristic :

$$\mu'_k(\mathrm{MTF}) \le \mu_k(\mathrm{MTF})$$

Again, the following example is taken directly from the paper

$$\mu'_2(\mathrm{MTF}) \le 1.20710 \text{ Opt}$$

$$\mu'_3(\mathrm{MTF}) \le 1.11843 \text{ Opt}$$

$$\mu'_4(\mathrm{MTF}) \le 1.08302 \text{ Opt}$$

The expected search cost for the batched k transposition heuristic is also shown to satisfy the following inequalities

$\mu'_k(TR) \leq \mu'_k(MTF)$ and $\mu'_{k+1}(TR) \leq \mu'_k(TR)$

for all distributions, $k \geq 1$

C. Convergence Rate of Self-Organizing Heuristics

The primary measure of effectiveness of a heuristic has been the asymptotic search cost. A natural concern is the rate of convergence of these heuristics to their asymptotic efficiencies. Bitner[1] shows that although the transposition rule is asymptotically better than move to front rule, the latter converges to its asymptotic value more rapidly. This observation suggests that the move to front heuristic may be preferable if fewer requests will be made to the list, or the probability of access keeps varying.

According to Bitner, the first measure of convergence is *overwork* , which is defined as the area between the cost curve and its asymptote (for details refer to [1]). The "steeper" the cost curve is, the smaller the overwork will be. The overwork for the conventional MTF rule is given as

$$Ov(MTF) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{(p_i-p_j)^2}{2(p_i+p_j)^2}$$

No general form has been found for the overwork for the transposition rule. However, there are good reasons to

believe that the transposition rule does converge more slowly. In the move to front rule, records with higher access probability rise quickly to the top of the list. In contrast, the transposition rule allows a record to rise one position at a time, thus decreasing the search cost more slowly. Through simulation, Bitner has shown that Ov(MTF) < Ov(TR) when the retrieval probabilities obey Zipf's distribution.

The second measure of convergence rate suggested by Bitner[1] is the number of requests, r, required for the total cost of one heuristic to be lower than that of the other. The total cost is obtained by summation of the search cost for the first r requests. This measure suggests that, under r requests, one heuristic may outperform the other, even though it has a higher asymptotic search cost. Some values of r are given in Table 1.3a at the end of this chapter.

In the remainder of this chapter, we shall study the convergence rate of the k in a row heuristics. Through numerical examples, we can show that the k in a row heuristics (MTF or TR) take longer time to reach their respective asymptotic search cost than the corresponding simple heuristics. This observation suggests that, although the k in a row heuristics (MTF or TR) have been shown to have lower asymptotic search cost than the corresponding simple heuristics(i.e., k=1), their slow rates of

convergence to the asymptotic value should not be overlooked.

Now we shall consider the overwork for the simple and batched k heuristics. Intuitively, one would expect, for $k > 1$, Ov(MTF) < Ov(simple k MTF) < Ov(batched k MTF), Ov(TR) < Ov(simple k TR) < Ov(batched k TR), Ov(simple k MTF) < Ov(simple k TR), and Ov(batched k MTF) < Ov(batched k TR). The reason behind this is that the heuristic that is less likely to change the previously established order decreases the search cost more slowly. Unfortunately, no general expression has been found for the overwork for the transposition heuristics and simple k MTF. However, the overwork for batched k MTF can be derived as follows :

THEOREM 1.1

$$\text{Ov (batched k MTF)} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k(p_i - p_j)(p_i^k - p_j^k)}{2(p_i^k - p_j^k)^2}$$

PROOF:

Assume that any initial arrangement of records in the list is equally likely, and the first request for retrieving a record is at t=0. Subsequent requests are at t=1,2,3,...

1. Prob(R_i is ahead of R_j at t requests, $0 \le t \le k-1$) = 1/2, since R_i has a 1/2 probability of being ahead of R_j in the initial ordering, and no rearrangement occurs when $t < k$.
2. Record R_i is ahead of R_j at t requests, $t \ge k$, under one of the following two conditions :

a. neither R_i nor R_j was requested consecutively k times in a batch and R_i was initially ahead of R_j.

$$\text{Prob} = (1/2)(1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor}$$

($\lfloor x \rfloor$ refers to the largest integer less than or equal to x)

b. the k most recent requests for R_i were in batch m, and R_j was not requested consecutively k times in a batch after m.

$$\text{Prob} = \sum_{m=1}^{\lfloor \frac{t}{k} \rfloor} p_i^k (1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor - m}$$

$$= p_i^k \left[\frac{1 - (1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor}}{p_i^k + p_j^k} \right]$$

$$= \frac{p_i^k}{p_i^k + p_j^k} - (1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor} \frac{p_i^k}{p_i^k + p_j^k}$$

Therefore, Prob(R_i is ahead of R_j at time t, $t \geq k$)

$$= \frac{p_i^k}{p_i^k + p_j^k} + (1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor} \left[\frac{1}{2} - \frac{p_i^k}{(p_i^k + p_j^k)} \right]$$

$$= \frac{p_i^k}{p_i^k + p_j^k} + (1 - p_i^k - p_j^k)^{\lfloor \frac{t}{k} \rfloor} \frac{(p_j^k - p_i^k)}{2\ (p_i^k + p_j^k)} \quad \ldots\ldots(1.1)$$

Since Equation (1.1) above reduces to 1/2 when $0 \leq t < k$, it is also the general expression for Prob(R_i is ahead of R_j at time t, $t \geq 0$).

Hence, if each initial list is equally likely, the search cost at time t

$$= \sum_{i=1}^{n} p_i \ \{ 1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \text{Prob}(R_j \text{ is ahead of } R_i \text{ at time } t)\}$$

$$= 1 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{p_i p_j{}^k + p_j p_i{}^k}{p_i{}^k + p_j{}^k}$$

$$+ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - p_i{}^k - p_j{}^k)^{\lfloor \frac{t}{k} \rfloor} \frac{p_i(p_i{}^k - p_j{}^k) + p_j(p_j{}^k - p_i{}^k)}{2\ (p_i{}^k + p_j{}^k)}$$

$$= 1 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{p_i p_j{}^k + p_j p_i{}^k}{p_i{}^k + p_j{}^k}$$

$$+ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - p_i{}^k - p_j{}^k)^{\lfloor \frac{t}{k} \rfloor} \frac{(p_i - p_j)(p_i{}^k - p_j{}^k)}{2\ (p_i{}^k + p_j{}^k)}$$

The first two terms of the above equation are the asymptotic search cost; hence the overwork at time t is given by the last term. Summing the overwork over $0 \leq t \leq \infty$ gives

Ov(batched k MTF)

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{(p_i - p_j)(p_i{}^k - p_j{}^k)}{2\ (p_i{}^k + p_j{}^k)} \sum_{t=0}^{\infty} (1 - p_i{}^k - p_j{}^k)^{\lfloor \frac{t}{k} \rfloor}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k\ (p_i - p_j)(p_i{}^k - p_j{}^k)}{2\ (p_i{}^k + p_j{}^k)^2} \qquad \blacksquare$$

THEOREM 1.2

Ov (batched k MTF) $\geq$ k Ov (simple MTF)

for $k > 1$, and not all p_i's are equal.

PROOF:

Ov (batched k MTF)

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k(p_i - p_j)(p_i^k - p_j^k)}{2(p_i^k + p_j^k)^2}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k(p_i - p_j)^2 \sum_{t=0}^{k-1} p_i^t p_j^{k-1-t}}{2(p_i^k + p_j^k)^2}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k\ (p_i - p_j)^2\ [\sum_{t=0}^{k-1} p_i^t p_j^{k-1-t}]\ (p_i + p_j)^2}{2\ (p_i^k + p_j^k)^2\ (p_i + p_j)^2}$$

$$\geq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{k(p_i - p_j)^2}{2(p_i + p_j)^2}$$

$\geq$ k Ov (simple MTF) ■

To demonstrate the convergence rates of the other heuristics, the overwork for a list of n elements under different self-organizing rules is approximated in the following manner.

The conventional method of analyzing the self organizing schemes is to model the different arrangements of the list as states in a Markov chain whose transition matrix is determined by the retrieval probabilities. Let $\mathrm{Prob}[\pi_i^{(t)}]$ denote the probability of the state π_i at time t, $\pi_i = [R_{i1} R_{i2} \ldots R_{in}]$, $1 \leq i_j \leq n$. The search cost at time t is then given by $\sum_i \left(\sum_{j=1}^{n} j p_{ij} \right) \mathrm{Prob}[\pi_i^{(t)}]$.

Initially, each of the n! possible arrangements is equally likely. For a particular state π_i, $\mathrm{Prob}[\pi_i^{(0)}] = 1/n!$, and successive distributions of the state $\pi_i^{(t)}$, for $t > 0$, can be obtained according to the following equations :

1. Simple MTF Heuristic

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t)}$$

$$= \; p_{i1} \{ \; \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t-1)} + \sum_{j=2}^{n} \mathrm{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{in}]^{(t-1)} \; \}$$

2. Simple TR Heuristics

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t)}$$

$$= \; p_{i1} \; \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t-1)} + \sum_{j=1}^{n-1} p_{ij} \; \mathrm{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{in}]^{(t-1)}$$

3. Simple k Heuristics

Let $(R_{i1}R_{i2}\ldots R_{im})_0$ be the initial state where records $R_{i1}, R_{i2}, \ldots, R_{im}$ are arranged in this order. In this state, a record R_j will be accessed with probability p_j, where $1 \le j \le n$, and a transition will be made from $(R_{i1}R_{i2}\ldots R_{im})_0$ to $(R_{i1}R_{i2}\ldots R_{im})_{j(1)}$. From $(R_{i1}R_{i2}\ldots R_{im})_{j(1)}$, accessing the same record R_j will cause a transition to $(R_{i1}R_{i2}\ldots R_{im})_{j(2)}$. Otherwise, state $(R_{i1}R_{i2}\ldots R_{im})_{x(1)}$ will be reached if record R_x is accessed, $x \ne j$. When the same record R_y is accessed k times in a row, the move to front/transposition rule is applied, and state $(\ldots)_0$ will be reached. In other words, the subscript y(z) for the state $(\pi_i)_{y(z)}$ is used to remember the number of consecutive requests for record R_y, when the current configuration is π_i. The stationary distribution of a particular arrangement, π_i, is given by the sum of the probabilities of states $(\pi_i)_0$, $(\pi_i)_{1(1)}$, ... $(\pi_i)_{1(k-1)}$, ... ,and $(\pi_i)_{n(k-1)}$.

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t)}$$

$$= \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{0}^{(t)} + \sum_{y=1}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{y(z)}^{(t)}$$

where,

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{x(1)}^{(t)}$$

$$= p_{x} \left\{ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{0}^{(t-1)} + \sum_{\substack{y=1 \\ y \neq x}}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{y(z)}^{(t-1)} \right\}$$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{x(z)}^{(t)} = p_{x} \, \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{x(z-1)}^{(t-1)}, \qquad 1 \le x \le n, \quad 2 \le z \le k-1$$

and

a) Simple k MTF

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{0}^{(t)}$$

$$= p_{i1} \left\{ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{i1(k-1)}^{(t-1)} + \sum_{j=2}^{n} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{in}]_{i1(k-1)}^{(t-1)} \right\}$$

b) Simple k TR

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{0}^{(t)}$$

$$= p_{i1} \, \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]_{i1(k-1)}^{(t-1)} + \sum_{j=1}^{n-1} p_{ij} \, \mathrm{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{in}]_{ij(k-1)}^{(t-1)}$$

4. Batched k Heuristics

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(t)} = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(z)}$$

where $z = k\lfloor \frac{t}{k} \rfloor$, and for $z > 0$

a) Batched k MTF

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(z)}$$
$$= \left(1 - \sum_{j=2}^{n} p_{ij}^{k} \right) \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(z-k)}$$
$$+ p_{i1}^{k} \sum_{j=2}^{n} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{in}]^{(z-k)}$$

b) Batched k TR

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(z)}$$
$$= \left(1 - \sum_{j=2}^{n} p_{ij}^{k} \right) \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{in}]^{(z-k)}$$
$$+ \sum_{j=1}^{n-1} p_{ij}^{k} \, \mathrm{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{in}]^{(z-k)}$$

The asymptotic distribution is approximated by the distribution when t is large enough, and the overwork can be obtained by summation of the difference between the search cost at each time instant and the asymptotic search cost.

Tables 1.1a, 1.1b, and 1.2 below show the expected search cost and the overwork for a list of n elements whose retrieval distribution is given by Zipf's distribution. The results obtained suggest that although the k in a row heuristics are asymptotically more efficient, they generally incur greater overwork as well. However, the study of

overwork alone is not a good measure of the rate of convergence; the asymptotic search cost should also be taken into consideration. Since overwork is defined in terms of the difference between search cost at each time instant and the asymptotic search cost, the heuristic with lower asymptotic search cost would generally result in larger overwork.

The second measure of convergence rate is the number of requests required for the total cost of one heuristic to be lower than that of the other. The measurement is approximated in a manner similar to the approximation of overwork. Tables 1.3a and 1.3b below summarize the results obtained.

From the study of the convergence rates, it may be seen that heuristics that are asymptotically more efficient may not be a preferable choice if few requests are made. Unfortunately, at present, there is no known method of deciding the optimal heuristic for the number of requests required.

D. Overview of Subsequent Chapters

In the next few chapters, we shall consider a linear search system with a fixed number of records, and a fixed-size list, whose size may be smaller than or equal to the number of records. Only a subset of records can be maintained in the list at one time. The frequency of

accessing each record is not known in advance. With some slight modifications, the self-organizing heuristics discussed so far can be adapted to improve performance.

The performance of the class of move to front heuristics is studied in detail. Through numerical examples, the class of transposition heuristics is compared to the move to front counterpart. The results obtained suggest that the transposition rule has lower asymptotic search cost than the corresponding move to front rule, and the k in a row heuristics again outperform the simple heuristics asymptotically.

An application of the modified scheme is in the construction of paging algorithm in memory management. While the commonly known paging algorithm, *Least Recently Used (LRU)* , corresponds to the modified simple move to front heuristic, the results obtained suggest that transposition rule or k in a row heuristics may be incorporated into the construction of the paging algorithms to improve performance.

TABLE 1.1a

The asymptotic search length for a list of n elements whose retrieval probabilities are given by Zipf's distribution, in terms of the number of searches through the list.

Heuristic	number of elements 3	4	5
optimal cost (optl)	1.6364	1.9200	2.1898
simple MTF rule	1.8545	2.2411	2.6104
simple TR rule	1.8182	2.1392	2.4303
simple 2 MTF rule	1.7754	2.1089	2.4227
simple 2 TR rule	1.7414	2.0358	2.3129
batched 2 MTF rule	1.7552	2.0842	2.3952
batched 2 TR rule	1.7248	2.0206	2.2991
simple 3 MTF rule	1.7180	2.0253	2.3158
simple 3 TR rule	1.6973	1.9871	2.2613
batched 3 MTF rule	1.7004	2.0071	2.2977
batched 3 TR rule	1.6850	1.9768	2.2523

TABLE 1.1b

The asymptotic search length for a list of n elements whose retrieval probabilities are given by Zipf's distribution, in terms of the optimal search cost.

Heuristic	number of elements 3	4	5
simple MTF rule	1.1333	1.1672	1.1921
simple TR rule	1.1111	1.1142	1.1098
simple 2 MTF rule	1.0849	1.0984	1.1064
simple 2 TR rule	1.0642	1.0603	1.0562
batched 2 MTF rule	1.0726	1.0855	1.0481
batched 2 TR rule	1.0540	1.0524	1.0499
simple 3 MTF rule	1.0499	1.0548	1.0575
simple 3 TR rule	1.0372	1.0349	1.0327
batched 3 MTF rule	1.0391	1.0454	1.0493
batched 3 TR rule	1.0297	1.0296	1.0285

TABLE 1.2

The overwork for a list of n elements whose retrieval probabilites are given by Zipf's distribution.

Heuristic	number of elements 3	4	5
simple MTF	0.2006	0.4463	0.7978
simple TR	0.4580	1.6505	3.9795
simple 2 MTF	1.0752	2.8137	5.7746
simple 2 TR	1.9893	6.9887	17.8381
batched 2 MTF	1.6454	4.4474	9.3983
batched 2 TR	2.8169	10.2260	27.3941
simple 3 MTF	3.8114	12.5235	31.7351
simple 3 TR	5.5799	23.6623	75.0309
batched 3 MTF	7.5675	27.6227	75.0761
batched 3 TR	10.1916	49.5195	171.4416

TABLE 1.3a

The number of requests required for simple/batched transposition rule to have lower cost than the corresponding simple/batched move-to-front rule, given that retrieval probabilities of the list of n elements satisfy Zipf's distribution.

Heuristic	3	number of elements 4	5	6
simple TR	6	10	15	20
simple 2 TR	25	51	94	161
batched 2 TR	36	78	156	280
simple 3 TR	83	248	658	1471
batched 3 TR	165	597	1740	4050

TABLE 1.3b

The number of requests required for simple/batched k rule (MTF or TR) to have lower cost than the corresponding simple/batched k-1 rule, assuming Zipf's distribution for the retrieval probabilities.

Heuristic	3	number of elements 4	5	6
simple 2 MTF	9	14	20	26
simple 3 MTF	38	83	167	308
simple 2 TR	17	46	104	217
simple 3 TR	74	309	1024	2763
batched 2 MTF	11	19	28	38
batched 3 MTF	83	213	475	918
batched 2 TR	21	62	153	338
batched 3 TR	168	824	2877	8010

II. Self-Organizing Linear Search Systems with Limited Buffers

So far, the study on self-organizing linear search has been limited to the sequential accessing of the same fixed number of records. However, this may not be a realistic representation of a sequentially access system. More often, one would not be dealing with the same list of records; additional records may be added to the system, or existing records are deleted.

Here, we shall relax the assumption that the same records are to be sequentially accessed. We consider the following scheme:

> Given a set of n records, and a linear list (buffer) of size m, where $m \le n$. Only m of the n records can be placed in the linear list at one time, according to one of the ${}_nP_m$ possible orderings. At each instant of time, one of the n records is requested. To retrieve the record, the list is examined sequentially, starting from the first position, until the required record is found or the end of the list is reached. If the requested record is not in the list, this record will be recalled from some auxiliary storage and be placed in the list. One of the records in the list will have to be dropped to make room for this new record, according to some predefined replacement rule.

Here, the performance measures of interest are the average

search length and the average number of fetches from the auxiliary storage.

A random arrangement would require, on the average, $(n-m)/n$ fetches from auxiliary storage, and $\frac{1}{n} + \frac{2}{n} + \ldots + \frac{m}{n} + \frac{m(n-m)}{n} = (m+2mn-m^2)/n$ searches through the buffer.

With some slight modifications, the conventional self-organizing heuristics (MTF or TR scheme) could be adapted to improve the performance. The modifications need to be made in order to accommodate the replacement rule. The schemes described below are basically the same as the conventional MTF and TR schemes, except when a requested record does not reside in the list. We shall adopt the same notation as before. Let $\pi_i(j)$ refer to the new position of record in position j, after a record in position i is requested. For records that are not in the list, assume that their position is "external".

Modified Move to Front Heuristic

for $1 \le i \le m$,

$$\pi_i(j) = \begin{cases} j+1, & 1 \le j \le i-1 \\ 1, & j = i \\ j, & i < j \le m \\ \text{external}, & j = \text{external} \end{cases}$$

for i = external,

$$\pi_i(j) = \begin{cases} j+1, & 1 \le j \le m-1 \\ \text{external}, & j = m \\ 1, & \text{for the requested record in external storage} \\ \text{external}, & \text{for the remaining records in external storage} \end{cases}$$

i.e., a requested record is always moved to position 1. If this requested record was originally in the array, say position i, then records in position 1 through i-1 are moved back one position. Otherwise, all records in array are moved back one position, forcing the last record to be dropped from the list.

Modified Transposition Heuristic

for $1 \le i \le m$,

$$\pi_i(j) = \begin{cases} j-1, & \text{if } j = i \text{ and } j \neq 1 \\ j+1, & \text{if } j = i-1 \\ j, & \text{otherwise (including records in external storage)} \end{cases}$$

for i = external,

$$\pi_i(j) = \begin{cases} m, & \text{for requested record in external storage} \\ \text{external}, & j = m \\ j, & \text{otherwise} \end{cases}$$

i.e., if a record does not reside in the list when requested, it will replace the last record in the array. Otherwise, it will be exchanged with the record immediately preceding it.

These schemata may not be truly representatives of systems involving insertions and deletions. However, the schemata are important for the following reasons. First, study of the performance with various buffer sizes demonstrates an analysis of the performance with varying amounts of insertions and deletions, since larger buffer size represents less insertion and deletion activities. Secondly, these schemata can be applied to the construction of paging algorithms for storage management. The *Least Recently Used (LRU)* strategy used in paging algorithms corresponds to the modified MTF heuristic here.

To study the performance of this system, we shall assume that each record has an independent probability p_i of being selected. The analysis of the performance involves modelling each of the ${}_nP_m$ possible arrangements of the buffer as a state in a Markov Chain. The one-step transition matrix is given by $[P_{ij}]$, which is determined by the p_i's. An account of the Markov Chain is given in Kemeny and Snell[9]. We shall be concerned with Markov chains that allow transitions from one state to any other states. Thus, the finite Markov chain is *irreducible*[1] and *aperiodic*[2]. This Markov chain is *ergodic*[3] and the limiting probabilities for each state,

$$\pi_j = \lim_{t\to\infty} \pi_j^{(t)} \qquad j = 1,\ldots \quad \pi_j > 0$$

always exist and are independent of the initial probability distribution. The limiting distribution $\pi = (\pi_1,\pi_2,\ldots)$ is given by the unique solution to the set of equations

$$\sum_i \pi_i = 1$$

$$\pi_j = \sum_i \pi_i P_{ij}, \qquad j = 1,2\ldots$$

Let $p(i,y)$ denote the long term probability that record R_i is in position y, and z_i the probability that R_i is not in the buffer.

---

[1] A chain is said to be irreducible if every state is reachable from every other state

[2] For every state i in an aperiodic chain, we can find two positive integers k1 and k2 such that the (k1)-step and (k2)-step transition probabilities, $P_{ii}^{(k1)} > 0$ and $P_{ii}^{(k2)} > 0$, and such that k1 and k2 have no common divisors other than 1.

[3] A finite-state Markov chain which is irreducible and aperiodic is ergodic

$$p(i,y) = \Sigma' \pi_j \quad (\text{summation of all those } \pi_j \text{ such that } R_i \text{ is in position } y)$$

$$z_i = 1 - \sum_{y=1}^{m} p(i,y)$$

The expected search length of a request, starting from position 1, is given by

$$\mu = \sum_{i=1}^{n} p_i \mu_i, \quad \text{where}$$

$$\mu_i = \left[\sum_{y=1}^{m} yp(i,y)\right] + mz_i$$

μ_i is the expected search length for record R_i.

The average number of fetches from the auxiliary storage when a record is requested is given by

$$\nu = \sum_{i=1}^{n} p_i z_i$$

Assume that $p_1 \geq p_2 \geq \ldots \geq p_n$. Then the optimal ordering would be to place the m most frequently accessed records in the buffer, according to the order $R_1R_2\ldots R_m$. The optimal expected search cost, μ(OPT), and the optimal number of fetches, ν(MTF), are given as :

$$\mu(\text{OPT}) = \sum_{i=1}^{m} ip_i + m \sum_{i=m+1}^{n} p_i$$

$$\nu(\text{OPT}) = \sum_{i=m+1}^{n} p_i$$

A. Simple MTF with Limited Buffer Size

The modified-MTF scheme corresponds to the LRU independent reference model (see Coffman and Denning[3]). Let Q_m be the set of all possible configurations of the m-size buffer, and π_i be a particular state, $\pi_i \epsilon Q_m$. Let π_i = $[R_{i1}R_{i2}\ldots R_{im}]$, where $1 \leq i_j \leq n$ and $R_{ij} \neq R_{ik}$ for $j \neq k$. For ease of reference, those records that are not in the buffer are labelled as $R_{i(m+1)}$ through R_{in}. Given that each record R_{ij} has a probability p_{ij} of being requested, the stationary distribution of the state π_i is given in [3]

THEOREM 2.1:

$$\mathrm{Prob}[\pi_i] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \prod_{j=1}^{m} \frac{p_{ij}}{\sum_{t=j}^{n} p_{it}}$$

Before proceeding to the proof, we shall present the following lemmas which indicate the two properties of the above expression for π_i.

Define :

$$\phi_z = \frac{\prod_{j=2}^{m} p_{ij}}{\left[\prod_{j=2}^{z} \left(p_{i1} + \sum_{t=j}^{n} p_{it}\right)\right]\left[\prod_{j=z+1}^{m} \left(\sum_{t=j}^{n} p_{it}\right)\right]} \quad , \quad 2 \leq z \leq m-1$$

and,

$$\phi_m = \frac{\prod_{j=2}^{m} p_{ij}}{\left[\prod_{j=2}^{m} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]}$$

LEMMA 2.1a:

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}] + \phi_{y+1} = \phi_y \ , \quad 2 \le y \le m-1$$

PROOF:

Case 1 : $2 \le y \le m-2$

$$\frac{\prod_{j=1}^{m} p_{ij}}{\left[\prod_{j=2}^{y+1} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it})\right]} + \phi_{y+1}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}\right]\left[p_{i1} + \sum_{t=y+1}^{n} p_{it}\right]}{\left[\prod_{j=2}^{y+1} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it})\right]}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}\right]\left[p_{i1} + \sum_{t=y+1}^{n} p_{it}\right]}{\left[\prod_{j=2}^{y} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]\left[p_{i1} + \sum_{t=y+1}^{n} p_{it}\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it})\right]}$$

$$= \phi_y$$

Case 2 : $y = m-1$

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{i(m-1)}R_{i1}R_{im}] + \phi_m$$

$$= \frac{\prod_{j=1}^{m} p_{ij}}{\left[\prod_{j=2}^{m} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]\left[\sum_{t=m}^{n} p_{it}\right]} + \frac{\prod_{j=2}^{m} p_{ij}}{\left[\prod_{j=2}^{m} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}\right]\left[p_{i1} + \sum_{t=m}^{n} p_{it}\right]}{\left[\prod_{j=2}^{m-1} (p_{i1} + \sum_{t=j}^{n} p_{it})\right]\left[p_{i1} + \sum_{t=m}^{n} p_{it}\right]\left[\sum_{t=m}^{n} p_{it}\right]}$$

$$= \phi_{m-1} \quad \blacksquare$$

Define $Prob_m(R_{i1}R_{i2}\ldots R_{im})$ to be the stationary probability of the state where records $R_{i1}, R_{i2}, \ldots, R_{im}$ are arranged accordingly, and the size of buffer is m. Again, records that are not in the buffer are labelled R_{m+1} through R_n.

LEMMA 2.1b:

$$Prob_m(R_{i1}R_{i2}\ldots R_{im}) = \sum_{x=m+1}^{n} Prob_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

This lemma indicates that a set of m records, $R_{i1}R_{i2}\ldots R_{im}$, has the same probability of occupying the first m positions in an (m+1)-size buffer as in an m-size buffer.

PROOF:

$$Prob_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix}) = \frac{\left[\prod_{j=1}^{m} p_{ij}\right] (p_{ix})}{\left[\prod_{j=1}^{m+1} (\sum_{t=j}^{n} p_{it})\right]}$$

$$\therefore \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}\right]\left[\sum_{x=m+1}^{n} p_{ix}\right]}{\left[\prod_{j=1}^{m}\left(\sum_{t=j}^{n} p_{it}\right)\right]\left[\sum_{t=m+1}^{n} p_{it}\right]}$$

$$= \mathrm{Prob}_{m}(R_{i1}R_{i2}\ldots R_{im}) \quad \blacksquare$$

To visualize the above lemma, one could also think of the n records as being arranged in an array of n entries, reorganized according to the rule of the conventional MTF scheme. Suppose we are interested in a m-size buffer. Records that occupy positions m+1 through n are to be thought of as residing in the auxiliary storage, so that a request for one of these records will necessitate m searches through the buffer and a fetch from the auxiliary storage. In other words, positions m+1 through n should be treated as the same position, namely "external". Notice that the modified-MTF scheme is satisfied : a requested record will always be moved to the top of the list, and requesting a record anywhere between m+1 and n (external) will force the record in position m to be moved to m+1 (i.e. external). Therefore, the stationary distribution of a configuration in a linear search system with an m-size buffer is given by the summation of all states in the conventional MTF n-element configuration that has the same sequence of m elements appearing in front.

PROOF OF THEOREM 2.1:

The equation given in Theorem 2.1 must satisfy the following expressions :

i) $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$

$$= p_{i1} \{ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] + \sum_{j=2}^{m} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \sum_{x=m+1}^{n} \mathrm{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \}$$

ii) $\sum_{i} \mathrm{Prob}[\pi_i] = 1$

Case 1 : to show that prob[π_i] satisfies (i) :

Rearranging (i) becomes

$$\frac{1 - p_{i1}}{p_{i1}} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \sum_{j=2}^{m} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \sum_{x=m+1}^{n} \mathrm{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \} \quad \ldots\ldots\ldots \quad (a)$$

Substituting prob[π_i] into L.H.S. of (a) gives

$$\frac{1 - p_{i1}}{p_{i1}} \left[\prod_{j=1}^{m} \frac{p_{ij}}{\sum_{t=j}^{n} p_{it}} \right] = \frac{\prod_{j=2}^{m} p_{ij}}{\left[\sum_{j=1}^{n} p_{ij} \right] \left[\prod_{j=3}^{m} \left(\sum_{t=j}^{n} p_{it} \right) \right]}$$

Substituting prob[π_i] into R.H.S. gives

$$\sum_{j=2}^{m-1} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \mathrm{Prob}[R_{i2}\ldots R_{im}R_{i1}]$$

$$+ \sum_{x=m+1}^{n} \frac{\left[\prod_{j=2}^{m} p_{ij} \right] (p_{ix})}{\left[\prod_{j=2}^{m+1} \left(p_{i1} + \sum_{t=j}^{n} p_{it} \right) \right]}$$

$$= \sum_{j=2}^{m-1} \text{Prob}[R_{i2} \ldots R_{ij} R_{i1} R_{i(j+1)} \ldots R_{im}]$$

$$+ \frac{\left[\prod_{j=2}^{m} p_{ij} \right] \left[p_{i1} + \sum_{x=m+1}^{n} p_{ix} \right]}{\left[\prod_{j=2}^{m} \left(p_{i1} + \sum_{t=j}^{n} p_{it} \right) \right] \left[p_{i1} + \sum_{t=m+1}^{n} p_{it} \right]}$$

$$= \sum_{j=2}^{m-1} \text{Prob}[R_{i2} \ldots R_{ij} R_{i1} R_{i(j+1)} \ldots R_{im}] + \prod_{j=2}^{m} \frac{p_{ij}}{\left[p_{i1} + \sum_{t=j}^{n} p_{it} \right]}$$

$$= \sum_{j=2}^{m-2} \text{Prob}[R_{i2} \ldots R_{ij} R_{i1} R_{i(j+1)} \ldots R_{im}]$$

$$+ \text{Prob}[R_{i2} \ldots R_{i(m-1)} R_{i1} R_{im}] + \phi_m$$

$$= \sum_{j=2}^{m-2} \text{Prob}[R_{i2} \ldots R_{ij} R_{i1} R_{i(j+1)} \ldots R_{im}] + \phi_{m-1}$$

(by applying Lemma 2.1)

$= \phi_2$ (by applying Lemma 2.1 m-3 times)

$$= \frac{\prod_{j=2}^{m} p_{ij}}{\left[\prod_{j=2}^{2} \left(p_{i1} + \sum_{t=j}^{n} p_{it} \right) \right] \left[\prod_{j=3}^{m} \left(\sum_{t=j}^{n} p_{it} \right) \right]}$$

$$= \frac{\prod_{j=2}^{m} p_{ij}}{\left[\sum_{j=1}^{n} p_{ij} \right] \left[\prod_{j=3}^{m} \left(\sum_{t=j}^{n} p_{it} \right) \right]}$$

∴ *L.H.S. = R.H.S.*

Case 2 : to show that $\sum_i \pi_i = 1$

First, we shall show that, asymptotically, record R_i has a probability p_i of being in position 1, for all buffer sizes. In other words, summation of stationary distributions for all configurations with record R_i in position 1 is equal to p_i.

<u>Proof by induction:</u>

True for buffer size m = 1,

$$\text{Prob}[R_i] = \frac{p_i}{\sum_{j=1}^{n} p_j} = p_i$$

Suppose this is true for m=k.

There are ${}_{n-1}P_{k-1}$ possible configurations that has record R_i in the first position. Let one of these configurations be denoted by $\pi_z(k)$, $z = [R_i R_{j2} \ldots R_{jk}]$, where $1 \le j_t \le n$ and $j_t \ne i$. Then,

$$\sum_z \text{Prob}[\pi_z(k)] = p_i$$

For m=k+1, there are ${}_{n-1}P_k$ possible configurations that have record R_i in the first position. Divide these ${}_{n-1}P_k$ possible configurations into ${}_{n-1}P_{k-1}$ groups of (n-k) configurations each. Within each group, the same k records appear in the first k positions, and in the same order. Denote one of these groups by $\pi_{z+}(k+1)$, where z+ is the set $\{ [R_i R_{j2} \ldots R_{jk} R_x], \mid 1 \le x \le n,\ x \ne i, j_2, j_3, \ldots, j_k \}$. For each z+, there is a corresponding z in Q_k such that the same k elements appear in the first k positions.

$$\text{Prob}[\pi_{z+}(k+1)] = \sum_x \text{Prob}_{k+1}(R_i R_{j2} \ldots R_{jk} R_x)$$

$$= \text{Prob}_k(R_i R_{j2} \ldots R_{jk}) \quad \text{(by Lemma 2.1b)}$$

Therefore,

$$\sum_{z+} \text{Prob}[\pi_{z+}(k+1)] = \sum_z \text{Prob}[\pi_z(k)] = p_i$$

By induction, each record R_i has a probability p_i of being in position 1, for all buffer sizes.

Hence,

$$\sum_i \pi_i = \sum_{i=1}^{n} \{ \Sigma \text{ Prob}[\pi \text{ with } R_i \text{ in position 1}] \}$$

$$= p_1 + p_2 + \ldots + p_n$$

$$= 1 \quad ■$$

The next two lemmas apply when Lemma 2.1b is satisfied.

LEMMA 2.2:

$$\Sigma \text{ Prob}_m(R_i \ldots) = \text{Prob}_1(R_i) \;, \quad 1 \le m \le n$$

Σ refers to the summation of states in Q_m that have record R_i in the first postion. This lemma states that the probability of having record R_i in position 1, for all buffer sizes, is the same as the probability of having R_i in a single-size buffer.

PROOF:

Follows directly from the proof of Theorem 2.1. ■

Let $\mu(m)$ be the expected search length to fetch a record when the buffer size is m, and $\nu(m)$ be the expected

number of fetches from auxiliary storage. Then

LEMMA 2.3:

$$\mu(m+1) = \mu(m) + \nu(m) \ , \ 0 \le m \le n-1$$

PROOF:

Case 1 : m = 0

Obviously, $\mu(0) = 0$, and $\nu(0) = 1$.

When the buffer size is one, one search through the buffer is needed to determine if the requested record is present or not. Hence, $\mu(1) = 1$.Therefore,

$$\mu(1) = \mu(0) + \nu(0)$$

Case 2 : m > 0

Let $\pi_i(m)$ be a particular configuration of a buffer of size m, $\pi_i(m) = [R_{i1}R_{i2}\ldots R_{im}]$, and let the records that are not in the buffer be denoted by $R_{i(m+1)},\ldots,R_{in}$.

$$\mu(m) = \sum_i \left(\sum_{t=1}^{m} tp_{it} + m \sum_{t=m+1}^{n} p_{it} \right) \mathrm{Prob}[\pi_i(m)]$$

$$\nu(m) = \sum_i \left\{ \left(\sum_{t=m+1}^{n} p_{it} \right) \mathrm{Prob}[\pi_i(m)] \right\}$$

Following the same argument as in the proof of Theorem 1, we could again group the possible configurations of an (m+1)-size buffer into ${}_nP_m$ groups, where all configurations in the group have the same m elements appearing in the first m positions. Denote this group by $\pi_{i+}(m+1)$, where π_{i+} is the set $\{ [R_{i1}R_{i2}\ldots R_{im}R_{ix}] \mid x \ne i_1,\ldots i_m \}$. For each π_{i+}, we could find a corresponding π_i in Q_m such that the first m elements are the same.

$$\mu(m+1) = \sum_{i+} \left(\sum_{t=1}^{m} t p_{it} + (m+1) \sum_{t=m+1}^{n} p_{it} \right) \mathrm{Prob}[\pi_{i+}(m+1)]$$

$$= \sum_{i} \left(\sum_{t=1}^{m} t p_{it} + m \sum_{t=m+1}^{n} p_{it} \right) \mathrm{Prob}[\pi_{i}(m)]$$

$$+ \sum_{i} \left(\sum_{t=m+1}^{n} p_{it} \right) \mathrm{Prob}[\pi_{i}(m)]$$

(by Lemma 2.1b)

$$= \mu(m) + \nu(m) \quad \blacksquare$$

Theorem 2.1 allows us to write a rather complicated expression for the performance of the modified scheme using the move to front heuristics. The retrieval cost, in terms of expected number of searches through the buffer and expected number of auxiliary memory fetches, is given in the next theorem.

<u>THEOREM 2.2</u>

Under the modified move to front heuristics, the expected number of auxiliary memory fetches for a linear search system with n elements and m buffers is given by

$$\nu(m) = \sum_{\pi_i \epsilon Q_m} [1 - \sum_{j=1}^{m} p_{ij}] \mathrm{Prob}(\pi_i), \qquad \text{and}$$

the expected number of buffer lookups is given by

$$\mu(m) = 1 + \sum_{x=1}^{m-1} \sum_{\pi_i \epsilon Q_x} [1 - \sum_{j=1}^{x} p_{ij}] \mathrm{Prob}(\pi_i),$$

where Q_z, $1 \le z \le n$, is the set of all possible configurations of the buffer whose size is z. Let $\pi_i \epsilon Q_z$, $\pi_i = [R_{i1} R_{i2} \ldots R_{iz}]$, $1 \le z \le n$, and $\mathrm{Prob}[\pi_i]$ is given by Theorem (2.1).

PROOF:

$$\nu(m) = \sum_{i=1}^{n} p_i z_i \,, \qquad \text{where } z_i \text{ is the probability that } R_i \text{ is not in buffer.}$$

$$= \sum_{\pi_i \epsilon Q_m} [1 - \sum_{j=1}^{m} p_{ij}] \, \text{Prob}(\pi_i)$$

By Lemma 2.3,

$$\mu(m) = \mu(m-1) + \nu(m-1)$$

$$= \mu(m-2) + \nu(m-2) + \nu(m-1)$$

$$= \mu(m-3) + \nu(m-3) + \nu(m-2) + \nu(m-1)$$

$$= \mu(0) + \nu(0) + \sum_{x=1}^{m-1} \nu(x)$$

Since $u(0) = 0$, and $v(0) = 1$,

$$\mu(m) = 1 + \sum_{x=1}^{m-1} \sum_{\pi_i \epsilon Q_x} [1 - \sum_{j=1}^{x} p_{ij}] \, \text{Prob}(\pi_i) \qquad \blacksquare$$

B. Simple TR with Limited Buffer Size

Similarly, this transposition rule can be modelled as a Markov chain. Let π_i be a particular state, $\pi_i = [R_{i1}R_{i2}\ldots R_{im}]$. The equations below describe the states of the Markov chain.

i) $\text{Prob}\,[R_{i1}R_{i2}\ldots R_{im}]$

$$= p_{i1}\,\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$+ \sum_{j=1}^{m-1} p_{ij}\,\text{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{im}]$$

$$+ p_{im} \sum_{x=m+1}^{n} \text{Prob}[R_{i1}R_{i2}\ldots R_{i(m-1)}R_{ix}]$$

ii) $\sum_{i} \text{Prob}[\pi_i] = 1$

Unlike MTF, the asymptotic distribution of the states of the above Markov chain is difficult to obtain. We fail to obtain a general expression for the retrieval cost of the modified scheme using the transposition rule. Although the exact retrieval cost cannot be determined here, observations suggest that, asymptotically, the transposition rule should be at least as efficient as the move to front heuristic, in terms of both the expected number of buffer lookups and the expected number of fetches.

Let b(i,j) and b'(i,j) denote the probability that record R_i precedes record R_j for the conventional move to front and the transposition heuristic respectively. Conventional refers to the self-organizing system where the buffer size is as big as the number of records. Rivest [13] shows that $b'(i,j) \geq b(i,j)$ when $p_i \geq p_j$. This observation suggests that under the transposition rule, records have a better chance of being arranged in a near optimal order than under the move to front rule. Let $\{A_i, i=1,...,n-1\}$ denote the spectrum of self-organizing heuristics in the conventional system. Under the A_i heuristic, every time a record is retrieved, it will be moved forward i positions (or to the front of the list, if the record was in a position less than i). A_1 corresponds to the transposition rule, while A_{n-1} corresponds to the move to front rule. Bitner[1] conjectures that the average search time for A_k is asymptotically better than that of A_{k+1}. With these previous observations in mind, we shall now proceed to examine the

performance of the modified scheme under the transposition rule.

Another way of looking at the modified transposition rule would be to treat the records as being arranged in an array with n entries. To study the behavior of the rule when the buffer size is m, positions m+1 through n should be treated as "external". When a record between position 1 and m is requested, it will be exchanged with the record preceding it. However, requesting a record that resides in position between m+1 and n would bring forward the record to position m, and hence, one of the following heuristics, A_1, A_2,..., A_{n-m}, is used when an "external" record is required. This observation suggests that the performance of the modified TR rule should acquire a characteristic that is a combination of the A_1, A_2, ... , A_{n-m}, heuristics. Also note that when the buffer size is 1, the modified TR rule is exactly the same as the corresponding modified move to front rule. When the buffer size is greater than 1, the modified TR rule would have a higher probability of achieving a near optimal ordering than the corresponding move to front rule, since a combination of A_1, A_2, ... , A_{n-m} rules should perform better than the A_{n-1}(MTF) rule alone. With this observation, it is suggested that the transposition rule should be at least as efficient asymptotically as the corresponding move to front heuristic, for all buffer sizes. Unfortunately, we have not been able to provide a proof of this conjecture.

Tables 2.2a, 2,2b, 2.4a and 2.4b show the performances of the modified system under move to front and transposition rule, for a list of n elements whose retrieval probabilities are given by the Zipf and Wedge distributions respectively. The Wedge distribution is closer to uniformity, $p_i = 2(n+1-i)/(n(n+1))$. The retrieval cost for the move to front rule is directly calculated from Theorem 2.2, while a good approximation of the cost for the transposition rule is obtained through calculating the probabilities of each state at successive times, t=0,1,2,..., in a similar manner as in Chapter 1. Appendix D further describes this approximation, and includes some simulation studies on these modified heuristics.

From Tables 2.2a, 2.2b , 2.4a and 2.4b, the results obtained do serve as numerical evidence for the suggestion that transposition rule is more efficient than the corresponding move to front rule. The simulation results in Appendix D also supports this conjecture. However, as mentioned before, the validity of this claim remains to be proven.

TABLE 2.1

The optimal retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by Zipf's Distribution :

number of records	retrieval probability of record (given by Zipf's distribution) 1	2	3	4	5	6
3	0.5455	0.2727	0.1818			
4	0.4800	0.2400	0.1600	0.1200		
5	0.4380	0.2190	0.1460	0.1095	0.0876	
6	0.4082	0.2041	0.1361	0.1020	0.0816	0.0680

The cost is in terms of the number of buffer lookups, μ, and the expected number of auxiliary memory fetches, ν.

number of records	buffer size	optimal retrieval cost
3	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.4545\ \nu$
	2	$1.4545\ \mu + 0.1818\ \nu$
	3	$1.6364\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5200\ \nu$
	2	$1.5200\ \mu + 0.2800\ \nu$
	3	$1.8000\ \mu + 0.1200\ \nu$
	4	$1.9200\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5620\ \nu$
	2	$1.5620\ \mu + 0.3431\ \nu$
	3	$1.9051\ \mu + 0.1971\ \nu$
	4	$2.1022\ \mu + 0.0876\ \nu$
	5	$2.1898\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5918\ \nu$
	2	$1.5918\ \mu + 0.3878\ \nu$
	3	$2.9796\ \mu + 0.2517\ \nu$
	4	$2.2313\ \mu + 0.1497\ \nu$
	5	$2.3810\ \mu + 0.0680\ \nu$
	6	$2.4490\ \mu + 0.0000\ \nu$

TABLE 2.2a

The asymptotic retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by Zipf's distribution. The cost is in terms of μ, the expected number of searches through the buffer, and ν, the expected number of fetches from the auxiliary storage.

number of records	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5950\ \nu$	$1.0000\ \mu + 0.5950\ \nu$
	2	$1.5950\ \mu + 0.2595\ \nu$	$1.5682\ \mu + 0.2500\ \nu$
	3	$1.8545\ \mu + 0.0000\ \nu$	$1.8182\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6720\ \nu$	$1.0000\ \mu + 0.6720\ \nu$
	2	$1.6720\ \mu + 0.3962\ \nu$	$1.6310\ \mu + 0.3765\ \nu$
	3	$2.0682\ \mu + 0.1730\ \nu$	$1.9787\ \mu + 0.1605\ \nu$
	4	$2.2411\ \mu + 0.0000\ \nu$	$2.1392\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.7193\ \nu$	$1.0000\ \mu + 0.7193\ \nu$
	2	$1.7193\ \mu + 0.4811\ \nu$	$1.6696\ \mu + 0.4359\ \nu$
	3	$2.2004\ \mu + 0.2844\ \nu$	$2.0777\ \mu + 0.2582\ \nu$
	4	$2.4848\ \mu + 0.1256\ \nu$	$2.3163\ \mu + 0.1141\ \nu$
	5	$2.6104\ \mu + 0.0000\ \nu$	$2.4304\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.7515\ \nu$	$1.0000\ \mu + 0.7515\ \nu$
	2	$1.7515\ \mu + 0.5395\ \nu$	$1.6960\ \mu + 0.5066\ \nu$
	3	$2.2910\ \mu + 0.3622\ \nu$	$2.1459\ \mu + 0.3246\ \nu$
	4	$2.6532\ \mu + 0.2161\ \nu$	$2.4387\ \mu + 0.1914\ \nu$
	5	$2.8693\ \mu + 0.0967\ \nu$	$2.6174\ \mu + 0.0869\ \nu$
	6	$2.9660\ \mu + 0.0000\ \nu$	$2.7043\ \mu + 0.0000\ \nu$
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.7751\ \nu$	
	2	$1.7751\ \mu + 0.5824\ \nu$	
	3	$2.3575\ \mu + 0.4198\ \nu$	
	4	$2.7773\ \mu + 0.2842\ \nu$	
	5	$3.0615\ \mu + 0.1715\ \nu$	
	6	$3.2330\ \mu + 0.0777\ \nu$	
	7	$3.3107\ \mu + 0.0000\ \nu$	

TABLE 2.2b
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by Zipf's distribution. The cost is in terms of the optimal number buffer lookups and the optimal number of auxiliary memory fetches.

number of records	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.3091\ \nu$	$1.0000\ \mu + 1.3091\ \nu$
	2	$1.0966\ \mu + 1.4274\ \nu$	$1.0782\ \mu + 1.3751\ \nu$
	3	$1.1333\ \mu$	$1.1111\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.2923\ \nu$	$1.0000\ \mu + 1.2923\ \nu$
	2	$1.1000\ \mu + 1.4150\ \nu$	$1.0730\ \mu + 1.3446\ \nu$
	3	$1.1490\ \mu + 1.4417\ \nu$	$1.0993\ \mu + 1.3375\ \nu$
	4	$1.1672\ \mu$	$1.1142\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.2799\ \nu$	$1.0000\ \mu + 1.2799\ \nu$
	2	$1.1007\ \mu + 1.4022\ \nu$	$1.0689\ \mu + 1.2705\ \nu$
	3	$1.1550\ \mu + 1.4429\ \nu$	$1.0906\ \mu + 1.3100\ \nu$
	4	$1.1820\ \mu + 1.4338\ \nu$	$1.1018\ \mu + 1.3025\ \nu$
	5	$1.1921\ \mu$	$1.1099\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.2699\ \nu$	
	2	$1.1003\ \mu + 1.3912\ \nu$	
	3	$1.1573\ \mu + 1.4390\ \nu$	
	4	$1.1891\ \mu + 1.4436\ \nu$	
	5	$1.2051\ \mu + 1.4221\ \nu$	
	6	$1.2111\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.2618\ \nu$	
	2	$1.0996\ \mu + 1.3817\ \nu$	
	3	$1.1580\ \mu + 1.4333\ \nu$	
	4	$1.1926\ \mu + 1.4463\ \nu$	
	5	$1.2123\ \mu + 1.4363\ \nu$	
	6	$1.2225\ \mu + 1.4102\ \nu$	
	7	$1.2263\ \mu$	

TABLE 2.3

The optimal retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by the Wedge distribution :

number of records	retrieval probability of record (given by Wedge distribution)					
	1	2	3	4	5	6
3	0.5000	0.3333	0.1667			
4	0.4000	0.3000	0.2000	0.1000		
5	0.3333	0.2667	0.2000	0.1333	0.0667	
6	0.2857	0.2381	0.1905	0.1429	0.0952	0.0476

The cost is in terms of the number of buffer lookups, μ, and the expected number of auxiliary memory fetches, ν.

number of records	buffer size	optimal retrieval cost
3	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5000\ \nu$
	2	$1.5000\ \mu + 0.1667\ \nu$
	3	$1.6667\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6000\ \nu$
	2	$1.6000\ \mu + 0.3000\ \nu$
	3	$1.9000\ \mu + 0.1000\ \nu$
	4	$2.0000\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6667\ \nu$
	2	$1.6667\ \mu + 0.4000\ \nu$
	3	$2.0667\ \mu + 0.2000\ \nu$
	4	$2.2667\ \mu + 0.0667\ \nu$
	5	$2.3333\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.7143\ \nu$
	2	$1.7143\ \mu + 0.4762\ \nu$
	3	$2.1095\ \mu + 0.2857\ \nu$
	4	$2.4762\ \mu + 0.1429\ \nu$
	5	$2.6190\ \mu + 0.0476\ \nu$
	6	$2.6667\ \mu + 0.0000\ \nu$

TABLE 2.4a

The asymptotic retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by the Wedge distribution. The cost is in terms of μ, the expected number of buffer lookups and ν, the expected number of auxiliary memory fetches.

number of records	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.6111 \nu$	$1.0000 \mu + 0.6111 \nu$
	2	$1.6111 \mu + 0.2611 \nu$	$1.5463 \mu + 0.2550 \nu$
	3	$1.8722 \mu + 0.0000 \nu$	$1.8403 \mu + 0.0000 \nu$
4	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.7000 \nu$	$1.0000 \mu + 0.7000 \nu$
	2	$1.7000 \mu + 0.4194 \nu$	$1.6770 \mu + 0.4040 \nu$
	3	$2.1194 \mu + 0.1734 \nu$	$2.0536 \mu + 0.1558 \nu$
	4	$2.2929 \mu + 0.0000 \nu$	$2.2094 \mu + 0.0000 \nu$
5	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.7556 \nu$	$1.0000 \mu + 0.7556 \nu$
	2	$1.7556 \mu + 0.5227 \nu$	$1.7338 \mu + 0.5064 \nu$
	3	$2.2783 \mu + 0.3077 \nu$	$2.2061 \mu + 0.2810 \nu$
	4	$2.5860 \mu + 0.1238 \nu$	$2.4623 \mu + 0.1047 \nu$
	5	$2.7098 \mu + 0.0000 \nu$	$2.5670 \mu + 0.0000 \nu$
6	0	1.0000ν	
	1	$1.0000 \mu + 0.7937 \nu$	
	2	$1.7937 \mu + 0.5950 \nu$	
	3	$2.3887 \mu + 0.4072 \nu$	
	4	$2.7959 \mu + 0.2359 \nu$	
	5	$3.0318 \mu + 0.0927 \nu$	
	6	$3.1246 \mu + 0.0000 \nu$	
7	0	1.0000ν	
	1	$1.0000 \mu + 0.8214 \nu$	
	2	$1.8214 \mu + 0.6484 \nu$	
	3	$2.4698 \mu + 0.4825 \nu$	
	4	$2.9523 \mu + 0.3269 \nu$	
	5	$3.2792 \mu + 0.1867 \nu$	
	6	$3.4660 \mu + 0.0720 \nu$	
	7	$3.5379 \mu + 0.0000 \nu$	

TABLE 2.4b
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n records with m buffers, whose retrieval probabilities are given by the Wedge distribution The cost is in terms of the optimal number of buffer lookups and optimal number of auxiliary memory fetches.

number of records	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.2222\ \nu$	$1.0000\ \mu + 1.2222\ \nu$
	2	$1.0741\ \mu + 1.5663\ \nu$	$1.0309\ \mu + 1.5297\ \nu$
	3	$1.1233\ \mu$	$1.1041\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1667\ \nu$	$1.0000\ \mu + 1.1667\ \nu$
	2	$1.0625\ \mu + 1.3980\ \nu$	$1.0481\ \mu + 1.3467\ \nu$
	3	$1.1155\ \mu + 1.7340\ \nu$	$1.0808\ \mu + 1.5580\ \nu$
	4	$1.1465\ \mu$	$1.1047\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1333\ \nu$	$1.0000\ \mu + 1.1333\ \nu$
	2	$1.0533\ \mu + 1.3068\ \nu$	$1.0403\ \mu + 1.2660\ \nu$
	3	$1.1024\ \mu + 1.5385\ \nu$	$1.0675\ \mu + 1.4050\ \nu$
	4	$1.1409\ \mu + 1.8561\ \nu$	$1.0863\ \mu + 1.5697\ \nu$
	5	$1.1614\ \mu$	$1.1002\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.1112\ \nu$	
	2	$1.0463\ \mu + 1.2495\ \nu$	
	3	$1.0905\ \mu + 1.4253\ \nu$	
	4	$1.1291\ \mu + 1.6508\ \nu$	
	5	$1.1576\ \mu + 1.9475\ \nu$	
	6	$1.1717\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0952\ \nu$	
	2	$1.0408\ \mu + 1.2104\ \nu$	
	3	$1.0805\ \mu + 1.3512\ \nu$	
	4	$1.1166\ \mu + 1.5254\ \nu$	
	5	$1.1477\ \mu + 1.7432\ \nu$	
	6	$1.1692\ \mu + 2.0168\ \nu$	
	7	$1.1793\ \mu$	

III. K in a Row Heuristics for Linear Search Systems with Limited Buffer Size

The notion of *k in a row* heuristic could again be introduced to the memory-free self-organizing rules for the scheme with limited buffer. The approach is similar to the one discussed in the conventional linear search system, i.e., the modified MTF (or TR) rule is applied only when the same element is accessed k times in a row. For this class of heuristics, $\log_2(nk)$ extra bits of storage are needed for a system with n elements.

In this chapter, the two k in a row heuristics, simple and batched k, are studied. We shall show that, the self-organizing heuristics for the linear search system with limited buffer size can perform more efficiently asymptotically, with the introduction of a few extra bits of storage. The asymptotic retrieval cost, in terms of both the expected number of buffer lookups and the expected number of auxiliary memory fetches, can be reduced when a few previous requests are "remembered".

The performance of the k in a row heuristics is again analyzed by modelling each of the ${}_nP_m$ possible configurations of the m buffers as a state in the Markov Chain. The class of move to front heuristics is studied in detail. While no general expression can be obtained for the performance of the transposition rule, numerical examples are provided to demonstrate the efficiency of the

transposition rule over the corresponding move to front rule.

Once again, we are looking at a self-organizing linear search system with n records and m buffers, where each record, R_i, has an independent probability p_i of being accessed.

A. Simple K Heuristic with Limited Buffer Size

The modified MTF (or TR) heuristic is applied only if the same element is accessed k times in a row.

Modified Simple K MTF

We first analyze the simple k heuristic with the modified move to front rule. The performance of this heuristic is analyzed in a manner analogous to that for the modified MTF heuristic. Let π_i be a particular state of the Markov chain, $\pi_i \in Q_m$. Let $\pi_i = [R_{i1}R_{i2}\ldots R_{im}]$, and let those records that are not in the buffer for this configuration be labelled as $R_{i(m+1)}$ through R_{in}. Each record R_{ij} has a corresponding probability p_{ij} of being accessed.

Let $(R_{i1}R_{i2}\ldots R_{in})_0$ be the initial state where records $R_{i1},R_{i2},\ldots,R_{in}$ are arranged in this order, and let $(R_{i1}R_{i2}\ldots R_{in})_{y(z)}$, $1 \le y \le n$ and $1 \le z \le k-1$, be the state where the z previous requests are all for record R_y.

The equations below describe the Markov chain :

i) $\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0$

$$= p_{i1} \{ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{i1(k-1)} + \sum_{j=2}^{m} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}]_{i1(k-1)} + \sum_{x=m+1}^{n} \text{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}]_{i1(k-1)} \}$$

where

$\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)}$

$$= p_j \{ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{\substack{y=1 \\ y \neq j}}^{n} \sum_{z=1}^{k-1} \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)} \}$$

$$\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} = p_j \, \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t-1)}, \quad 2 \le t \le k-1$$

$$\text{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{y=1}^{n} \sum_{z=1}^{k-1} \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)}$$

ii) $\sum_{i} \text{Prob}[\pi_i] = 1$

For ease of notation, we shall denote $\sum_{t=1}^{k-1} p_j^{\,t}$ by X_j for the discussion that follows.

THEOREM 3.1:

Under the modified simple k MTF heuristic, the stationary distribution of the state π_i is given by

$$\mathrm{Prob}[\pi_i] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \prod_{j=1}^{m} \frac{\left[\, p_{ij}{}^{k} \prod_{t=j+1}^{n} (1+X_{it}) \,\right]}{\left[\, \sum_{x=j}^{n} \left\{ p_{ix}{}^{k} + \prod_{\substack{t=j \\ t\neq x}}^{n} (1+X_{it}) \right\} \,\right]}$$

PROOF: See Appendix B.

In Appendix B, we show that

$$\mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im}) = \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

The equation above allows us to write down

$$\mu(m+1) = \mu(m) + \nu(m), \quad 0 \leq m \leq n-1 \qquad \ldots\ldots(3.1)$$

as in Lemma 2.2.

With Theorem 3.1 and Equation (3.1) above, we can now write a rather complicated expression for the retreival cost for this heuristic. The performance is given in the next theorem.

THEOREM 3.2

For a limited-buffer linear search system that adapts the modified simple k heuristics, the expected number of

auxiliary memory fetches is given by

$$\nu(m) = \sum_{\pi_i \epsilon Q_m} [1 - \sum_{j=1}^{m} p_{ij}] \text{Prob}[\pi_i], \qquad \text{and}$$

the expected number of buffer lookups is given by

$$\mu(m) = 1 + \sum_{x=1}^{m-1} \sum_{\pi_i \epsilon Q_x} [1 - \sum_{j=1}^{x} p_{ij}] \text{Prob}[\pi_i],$$

where Q_z is the set of all configurations of the buffer when the buffer size is z, and $\pi_i \epsilon Q_z$ is a particular state, $\pi_i = [R_{i1}R_{i2}...R_{iz}]$. $\text{Prob}[\pi_i]$ is given by Theorem 3.1.

PROOF:

This is equivalent to Theorem 2.2.

To study how well this simple k MTF heuristic performs as compared to the corresponding simple MTF rule, we first take a look at the expected number of auxiliary memory fetches. Since we have shown that the expected number of buffer lookups, $\mu(m)$, is given by $\nu(0) + \nu(1) + ... + \nu(m-1)$, a lower number of expected auxiliary memory fetches, for all buffer sizes, will directly result in lower expected number of buffer lookups as well.

Let p(i,j) be the probability of record R_i in position j. The expected number of auxiliary memory fetches is shown to be :

$$\nu(m) = \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \{ (p_i - p_{m+1}) [1 - \sum_{j=1}^{m} p(i,j)] \}$$

$$+ \sum_{i=m+2}^{n} \{ (p_{m+1} - p_i) \sum_{j=1}^{m} p(i,j) \} \qquad .....(3.2)$$

(See Appendix A)

Without loss of generality, assume the records are labelled in non-increasing order of retrieval probability. From Equation (3.2) it is obvious that the smallest value for $\nu(m)$ is $\Sigma_{i=m+1}^{n}$ p_i, and this is achieved when the (n-m) smallest records are not in the buffer.

We have shown earlier that for the class of move to front heuristics, the n records can be treated as being arranged in a list, and reorganizing occurs in the same way as in the corresponding conventional system. When the buffer size of interest is m, positions m+1 through n would be treated as "external", and the distribution of the configuration could be preserved. Tables 3.1 and 3.2 below show some values of $p_k(i,j)$, i.e., probability of record R_i in position j under the simple k heuristics for the conventional system, where the buffer size is as big as the number of records. When the buffer size is m, $p_k(i,j)$ is the same as in the conventional system, for $1 \leq j \leq m$. The probability of record R_i not being in the buffer is then given by summation of $p_k(i,j)$ in the conventional system, for j ranging from m+1 to n. Tables 3.1 and 3.2 illustrate how the $p_k(i,j)$ changes with k. We can see from the two tables that, as k increases, simple k heuristics tend to arrange the records more and more closely to the optimal arrangement, i.e., record R_i has a higher probability of being arranged closer to its optimal position, i, as k

increases. Previous study by Gonnet, Munro and Suwanda[5] has shown that, under the conventional simple k heuristics, $b_k(i,j) \geq b_{k-1}(i,j)$ when $p_i \geq p_j$. $b_k(i,j)$ refers to the probability that record R_i is in front of R_j. This observation suggests that the simple k heuristic attempts to rearrange the records more closely to the optimal arrangment, as k increases.

Unfortunately, no concise general expression is obtained for $p_k(i,j)$ for this simple k heuristic, and thus the cost for various k cannot be compared directly. We then have to examine the retrieval cost for specific distributions. The retrieval cost for a list of n elements, assuming the retrieval probabilities satisfy Zipf's distribution, is calculated according to Theorem 3.2, and is given in Tables 3.3a and 3.4a. The ratio of the retrieval cost to the optimal cost is presented in Tables 3.3b and 3.4b respectively. Tables 3.5a, 3.5b, 3.6a and 3.6b illustrate the retrieval cost for a list of elements satisfying the Wedge distribution. Numerical examples presented show that the simple k heuristic has a lower number of memory fetches than the corresponding k-1 heuristic, for all buffer sizes. Consequently, the expected number of buffer lookups for the simple k heuristic decreases as k increases, for all buffer sizes. Thus, at the expense of a small amount of extra memory, $\log_2(n)+\log_2(k)$ bits to be exact, the modified simple k heuristics can be asymptotically more efficient than that of the corresponding

simple move to front heuristic.

Modified Simple K TR

The Markov chain used to model this transposition heuristic is given by the following equations :

i) $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0$

$$= p_{i1} \; \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{i1(k-1)}$$

$$+ \sum_{j=1}^{m-1} p_{ij} \; \mathrm{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{im}]_{ij(k-1)}$$

$$+ p_{im} \sum_{x=m+1}^{n} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{i(m-1)}R_{ix}]_{im(k-1)}$$

where

$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)}$

$$= p_j \left\{ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{\substack{y=1 \\ y \neq j}}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)} \right\}$$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} = p_j \; \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t-1)}, \quad 2 \leq t \leq k-1$$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{y=1}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)}$$

ii) $\sum_i \mathrm{Prob}[\pi_i] = 1$

No general expression is obtained for the solution of the above Markov chain, and the exact cost for the transposition rule is still unknown. We can only

demonstrate, through numerical examples and simulation results, that the transposition rule can perform better asymptotically than the corresponding simple k move to front rule, in terms of both the expected number of buffer lookups and the expected number of auxiliary memory fetches. A good approximation of the retrieval cost, for a list of n elements with m buffers, is obtained in an analogous manner as in Chapter 2. Tables 3.3a, 3.3b, 3.4a, 3.4b, 3.5a, 3.5b, 3.6a, and 3.6b illustrate some values of the retrieval cost. Appendix D includes some simulation results.

TABLE 3.1

The probability of record i in position j, assuming retrieval probabilities satisfy Zipf's distribution, when the simple k MTF heuristic is adapted.

record i	probability of record i in position j					
	1	2	3	4	5	6
n = 4						
	Simple MTF rule :					
1	0.4800	0.3085	0.1597	0.0518		
2	0.2400	0.3000	0.2800	0.1800		
3	0.1600	0.2200	0.3014	0.3186		
4	0.1200	0.1715	0.2589	0.4496		
	Simple 2 MTF rule :					
1	0.6567	0.2651	0.0694	0.0088		
2	0.1960	0.4062	0.2907	0.1072		
3	0.0931	0.2061	0.3866	0.3142		
4	0.0542	0.1225	0.2534	0.5698		
	Simple 3 MTF rule :					
1	0.8053	0.1750	0.0189	0.0008		
2	0.1327	0.5574	0.2587	0.0513		
3	0.0430	0.1854	0.4897	0.2819		
4	0.0190	0.0822	0.2327	0.6661		
n = 6						
	Simple 1 MTF rule :					
1	0.4082	0.2814	0.1738	0.0913	0.0367	0.0086
2	0.2041	0.2291	0.2178	0.1777	0.1185	0.0527
3	0.1361	0.1662	0.1938	0.2026	0.1814	0.1199
4	0.1020	0.1291	0.1613	0.1952	0.2165	0.1958
5	0.0816	0.1053	0.1361	0.1760	0.2275	0.2734
6	0.0680	0.0888	0.1172	0.1571	0.2193	0.3495
	Simple 2 MTF rule :					
1	0.6255	0.2676	0.0851	0.0189	0.0026	0.0002
2	0.1829	0.3428	0.2677	0.1439	0.0526	0.0102
3	0.0862	0.1726	0.2687	0.2505	0.1613	0.0607
4	0.0500	0.1022	0.1745	0.2540	0.2564	0.1629
5	0.0326	0.0673	0.1187	0.1903	0.2881	0.3030
6	0.0229	0.0475	0.0853	0.1422	0.2389	0.4631
	Simple 3 MTF rule :					
1	0.8006	0.1760	0.0218	0.0015	0.0001	0.0000
2	0.1265	0.5175	0.2630	0.0783	0.0136	0.0011
3	0.0404	0.1696	0.3808	0.2720	0.1137	0.0234
4	0.0177	0.0747	0.1804	0.3328	0.2763	0.1182
5	0.0093	0.0392	0.0966	0.1957	0.3553	0.3038
6	0.0054	0.0230	0.0573	0.1197	0.2411	0.5535

TABLE 3.2

The probability of record i in position j, assuming retrieval probabilities satisfy Wedge distribution. Simple k MTF heuristic is adapted.

record	probability of record i in position j					
i	1	2	3	4	5	6
n = 4						
	Simple MTF rule :					
1	0.4000	0.3159	0.2063	0.0778		
2	0.3000	0.3083	0.2619	0.1298		
3	0.2000	0.2413	0.3175	0.2413		
4	0.1000	0.1345	0.2143	0.5512		
	Simple 2 MTF rule:					
1	0.5058	0.3322	0.1414	0.0206		
2	0.3064	0.3795	0.2579	0.0562		
3	0.1475	0.2224	0.4492	0.1809		
4	0.0402	0.0659	0.1515	0.7424		
	Simple 3 MTF rule :					
1	0.6051	0.3147	0.0766	0.0036		
2	0.2865	0.4729	0.2224	0.0182		
3	0.0952	0.1853	0.6067	0.1129		
4	0.0133	0.0271	0.0943	0.8653		
n = 6						
	Simple MTF rule :					
1	0.2857	0.2485	0.2038	0.1496	0.0860	0.0264
2	0.2381	0.2279	0.2077	0.1719	0.1136	0.0408
3	0.1905	0.1970	0.2014	0.1929	0.1519	0.0662
4	0.1429	0.1576	0.1773	0.2034	0.2035	0.1153
5	0.0952	0.1109	0.1347	0.1757	0.2603	0.2231
6	0.0476	0.0581	0.0749	0.1065	0.1846	0.5282
	Simple 2 MTF rule :					
1	0.3778	0.2946	0.1984	0.0994	0.0275	0.0023
2	0.2724	0.2758	0.2379	0.1531	0.0547	0.0059
3	0.1813	0.2113	0.2522	0.2269	0.1119	0.0163
4	0.1063	0.1347	0.1863	0.2926	0.2302	0.0499
5	0.0493	0.0658	0.0979	0.1764	0.4360	0.1747
6	0.0129	0.0177	0.0273	0.0516	0.1396	0.7510
	Simple 3 MTF rule :					
1	0.4672	0.3180	0.1621	0.0474	0.0052	0.0001
2	0.2855	0.3306	0.2557	0.1101	0.0176	0.0005
3	0.1543	0.2122	0.3301	0.2397	0.0611	0.0026
4	0.0687	0.1019	0.1821	0.4236	0.2082	0.0155
5	0.0214	0.0329	0.0616	0.1575	0.6187	0.1079
6	0.0028	0.0044	0.0083	0.0217	0.0893	0.8735

TABLE 3.3a : Simple 2 Heuristic (Zipf's Distribution)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by Zipf's distribution. The cost is in terms of μ, the expected number of searches through the buffer, and ν, the expected number of fetches from the auxiliary storage.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	 Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5482\ \nu$	$1.0000\ \mu + 0.5482\ \nu$
	2	$1.5482\ \mu + 0.2273\ \nu$	$1.5212\ \mu + 0.2202\ \nu$
	3	$1.7754\ \mu + 0.0000\ \nu$	$1.7414\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6163\ \nu$	$1.0000\ \mu + 0.6163\ \nu$
	2	$1.6163\ \mu + 0.3439\ \nu$	$1.5776\ \mu + 0.3311\ \nu$
	3	$1.9603\ \mu + 0.1486\ \nu$	$1.8934\ \mu + 0.1424\ \nu$
	4	$2.1089\ \mu + 0.0000\ \nu$	$2.0358\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6580\ \nu$	$1.0000\ \mu + 0.6580\ \nu$
	2	$1.6580\ \mu + 0.4156\ \nu$	$1.6134\ \mu + 0.3992\ \nu$
	3	$2.0735\ \mu + 0.2414\ \nu$	$1.9900\ \mu + 0.2294\ \nu$
	4	$2.3150\ \mu + 0.1078\ \nu$	$2.2103\ \mu + 0.1026\ \nu$
	5	$2.4227\ \mu + 0.0000\ \nu$	$2.3129\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.6863\ \nu$	
	2	$1.6863\ \mu + 0.4645\ \nu$	
	3	$2.1508\ \mu + 0.3053\ \nu$	
	4	$2.4561\ \mu + 0.1830\ \nu$	
	5	$2.6391\ \mu + 0.0833\ \nu$	
	6	$2.7223\ \mu + 0.0000\ \nu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.7071\ \nu$	
	2	$1.7071\ \mu + 0.5004\ \nu$	
	3	$2.2075\ \mu + 0.3522\ \nu$	
	4	$2.5596\ \mu + 0.2385\ \nu$	
	5	$2.7981\ \mu + 0.1457\ \nu$	
	6	$2.9438\ \mu + 0.0671\ \nu$	
	7	$3.0109\ \mu + 0.0000\ \nu$	

TABLE 3.3b : Simple 2 Heuristic (Zipf's Distribution)
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by Zipf's distribution. The cost is in terms of the optimal number of buffer lookups and optimal number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	 Transposition rule
3	0	$1.0000\,\nu$	$1.0000\,\nu$
	1	$1.0000\,\mu + 1.2062\,\nu$	$1.0000\,\mu + 1.2062\,\nu$
	2	$1.0644\,\mu + 1.2503\,\nu$	$1.0459\,\mu + 1.2112\,\nu$
	3	$1.0849\,\mu$	$1.0642\,\mu$
4	0	$1.0000\,\nu$	$1.0000\,\nu$
	1	$1.0000\,\mu + 1.1852\,\nu$	$1.0000\,\mu + 1.1852\,\nu$
	2	$1.0634\,\mu + 1.2282\,\nu$	$1.0379\,\mu + 1.1825\,\nu$
	3	$1.0891\,\mu + 1.2383\,\nu$	$1.0519\,\mu + 1.1867\,\nu$
	4	$1.0984\,\mu$	$1.0603\,\mu$
5	0	$1.0000\,\nu$	$1.0000\,\nu$
	1	$1.0000\,\mu + 1.1708\,\nu$	$1.0000\,\mu + 1.1566\,\nu$
	2	$1.0615\,\mu + 1.2113\,\nu$	$1.0329\,\mu + 1.1635\,\nu$
	3	$1.0884\,\mu + 1.2248\,\nu$	$1.0446\,\mu + 1.1639\,\nu$
	4	$1.1012\,\mu + 1.2306\,\nu$	$1.0514\,\mu + 1.1712\,\nu$
	5	$1.1064\,\mu$	$1.0562\,\mu$
6	0	$1.0000\,\nu$	
	1	$1.0000\,\mu + 1.1597\,\nu$	
	2	$1.0594\,\mu + 1.1978\,\nu$	
	3	$1.0865\,\mu + 1.2130\,\nu$	
	4	$1.1007\,\mu + 1.2224\,\nu$	
	5	$1.1084\,\mu + 1.2225\,\nu$	
	6	$1.1116\,\mu$	
7	0	$1.0000\,\nu$	
	1	$1.0000\,\mu + 1.1511\,\nu$	
	2	$1.0575\,\mu + 1.1872\,\nu$	
	3	$1.0843\,\mu + 1.2025\,\nu$	
	4	$1.0992\,\mu + 1.2137\,\nu$	
	5	$1.1080\,\mu + 1.2203\,\nu$	
	6	$1.1131\,\mu + 1.2178\,\nu$	
	7	$1.1153\,\mu$	

TABLE 3.4a : Simple 3 Heuristic (Zipf's Distribution)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probability is given by Zipf's distribution.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 0.5092\nu$	$1.0000\mu + 0.5092\nu$
	2	$1.5092\mu + 0.2088\nu$	$1.4919\mu + 0.2055\nu$
	3	$1.7180\mu + 0.0000\nu$	$1.6973\mu + 0.0000\nu$
4	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 0.5724\nu$	$1.0000\mu + 0.5724\nu$
	2	$1.5724\mu + 0.3151\nu$	$1.5503\mu + 0.3102\nu$
	3	$1.8876\mu + 0.1377\nu$	$1.8525\mu + 0.1347\nu$
	4	$2.0253\mu + 0.0000\nu$	$1.9871\mu + 0.0000\nu$
5	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 0.6117\nu$	$1.0000\mu + 0.6117\nu$
	2	$1.6117\mu + 0.3808\nu$	$1.5884\mu + 0.3753\nu$
	3	$1.9925\mu + 0.2229\nu$	$1.9519\mu + 0.2175\nu$
	4	$2.2154\mu + 0.1004\nu$	$2.1636\mu + 0.0978\nu$
	5	$2.3158\mu + 0.0000\nu$	$2.2614\mu + 0.0000\nu$
6	0	1.0000ν	
	1	$1.0000\mu + 0.6390\nu$	
	2	$1.6390\mu + 0.4260\nu$	
	3	$2.0650\mu + 0.2815\nu$	
	4	$2.3465\mu + 0.1698\nu$	
	5	$2.5162\mu + 0.0779\nu$	
	6	$2.5942\mu + 0.0000\nu$	
7	0	1.0000ν	
	1	$1.0000\mu + 0.6593\nu$	
	2	$1.6593\mu + 0.4596\nu$	
	3	$2.1188\mu + 0.3246\nu$	
	4	$2.4434\mu + 0.2208\nu$	
	5	$2.6643\mu + 0.1357\nu$	
	6	$2.7999\mu + 0.0630\nu$	
	7	$2.8629\mu + 0.0000\nu$	

TABLE 3.4b : Simple 3 Heuristic (Zipf's Distribution)
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probability is given by Zipf's distribution. The cost is in terms of the optimal number of buffer lookups and memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1204\ \nu$	$1.0000\ \mu + 1.1204\ \nu$
	2	$1.0376\ \mu + 1.1485\ \nu$	$1.0257\ \mu + 1.1304\ \nu$
	3	$1.0499\ \mu$	$1.0372\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1008\ \nu$	$1.0000\ \mu + 1.1008\ \nu$
	2	$1.0345\ \mu + 1.1254\ \nu$	$1.0199\ \mu + 1.1079\ \nu$
	3	$1.0487\ \mu + 1.1475\ \nu$	$1.0292\ \mu + 1.1225\ \nu$
	4	$1.0548\ \mu$	$1.0349\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.0884\ \nu$	$1.0000\ \mu + 1.0884\ \nu$
	2	$1.0318\ \mu + 1.1099\ \nu$	$1.0169\ \mu + 1.0939\ \nu$
	3	$1.0459\ \mu + 1.1309\ \nu$	$1.0246\ \mu + 1.1035\ \nu$
	4	$1.0538\ \mu + 1.1461\ \nu$	$1.0292\ \mu + 1.1164\ \nu$
	5	$1.0575\ \mu$	$1.0327\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0798\ \nu$	
	2	$1.0297\ \mu + 1.0985\ \nu$	
	3	$1.0431\ \mu + 1.1184\ \nu$	
	4	$1.0516\ \mu + 1.1343\ \nu$	
	5	$1.0568\ \mu + 1.1456\ \nu$	
	6	$1.0593\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0733\ \nu$	
	2	$1.0279\ \mu + 1.0904\ \nu$	
	3	$1.0408\ \mu + 1.1082\ \nu$	
	4	$1.0493\ \mu + 1.1287\ \nu$	
	5	$1.0550\ \mu + 1.1365\ \nu$	
	6	$1.0587\ \mu + 1.1434\ \nu$	
	7	$1.0605\ \mu$	

Table 3.5a : Simple 2 Heuristic (Wedge Distribution)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by the Wedge distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number of memory fetches.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5797\ \nu$	$1.0000\ \mu + 0.5797\ \nu$
	2	$1.5797\ \mu + 0.2212\ \nu$	$1.5603\ \mu + 0.2110\ \nu$
	3	$1.8009\ \mu + 0.0000\ \nu$	$1.7713\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6722\ \nu$	$1.0000\ \mu + 0.6772\ \nu$
	2	$1.6722\ \mu + 0.3744\ \nu$	$1.6505\ \mu + 0.3611\ \nu$
	3	$2.0466\ \mu + 0.1355\ \nu$	$1.9952\ \mu + 0.1249\ \nu$
	4	$2.1821\ \mu + 0.0000\ \nu$	$2.1201\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.7312\ \nu$	$1.0000\ \mu + 0.7312\ \nu$
	2	$1.7312\ \mu + 0.4799\ \nu$	$1.7104\ \mu + 0.4656\ \nu$
	3	$2.2111\ \mu + 0.2588\ \nu$	$2.1519\ \mu + 0.2408\ \nu$
	4	$2.4699\ \mu + 0.0907\ \nu$	$2.3820\ \mu + 0.0825\ \nu$
	5	$2.5606\ \mu + 0.0000\ \nu$	$2.4646\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.7722\ \nu$	
	2	$1.7722\ \mu + 0.5557\ \nu$	
	3	$2.3279\ \mu + 0.3571\ \nu$	
	4	$2.6850\ \mu + 0.1879\ \nu$	
	5	$2.8729\ \mu + 0.0647\ \nu$	
	6	$2.9376\ \mu + 0.0000\ \nu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.8023\ \nu$	
	2	$1.8023\ \mu + 0.6125\ \nu$	
	3	$2.4148\ \mu + 0.4344\ \nu$	
	4	$2.8492\ \mu + 0.2743\ \nu$	
	5	$3.1235\ \mu + 0.1419\ \nu$	
	6	$3.2645\ \mu + 0.0484\ \nu$	
	7	$3.3138\ \mu + 0.0000\ \nu$	

Table 3.5b : Simple 2 Heuristic (Wedge Distribution)
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by the Wedge distribution. The cost is in terms of the optimal number of buffer lookups and memory fetches.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 1.1594 \nu$	$1.0000 \mu + 1.1594 \nu$
	2	$1.0531 \mu + 1.3269 \nu$	$1.0402 \mu + 1.2657 \nu$
	3	1.0805μ	1.0628μ
4	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 1.1287 \nu$	$1.0000 \mu + 1.1287 \nu$
	2	$1.0451 \mu + 1.2480 \nu$	$1.0316 \mu + 1.2037 \nu$
	3	$1.0772 \mu + 1.3550 \nu$	$1.0501 \mu + 1.2490 \nu$
	4	1.0910μ	1.0601μ
5	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 1.0967 \nu$	$1.0000 \mu + 1.0967 \nu$
	2	$1.0387 \mu + 1.1999 \nu$	$1.0262 \mu + 1.1640 \nu$
	3	$1.0699 \mu + 1.2940 \nu$	$1.0412 \mu + 1.2040 \nu$
	4	$1.0896 \mu + 1.3598 \nu$	$1.0509 \mu + 1.2369 \nu$
	5	1.0974μ	1.0563μ
6	0	1.0000ν	
	1	$1.0000 \mu + 1.0811 \nu$	
	2	$1.0338 \mu + 1.1669 \nu$	
	3	$1.0627 \mu + 1.2499 \nu$	
	4	$1.0843 \mu + 1.3149 \nu$	
	5	$1.0969 \mu + 1.3592 \nu$	
	6	1.1016μ	
7	0	1.0000ν	
	1	$1.0000 \mu + 1.0697 \nu$	
	2	$1.0299 \mu + 1.1434 \nu$	
	3	$1.0565 \mu + 1.2165 \nu$	
	4	$1.0781 \mu + 1.2800 \nu$	
	5	$1.0932 \mu + 1.3249 \nu$	
	6	$1.1016 \mu + 1.3557 \nu$	
	7	1.1046μ	

TABLE 3.6a : Simple 3 Heuristic (Wedge Distribution)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by the Wedge distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number of memory fetches.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5551\ \nu$	$1.0000\ \mu + 0.5551\ \nu$
	2	$1.5551\ \mu + 0.1946\ \nu$	$1.5433\ \mu + 0.1888\ \nu$
	3	$1.7497\ \mu + 0.0000\ \nu$	$1.7321\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6517\ \nu$	$1.0000\ \mu + 0.6517\ \nu$
	2	$1.6517\ \mu + 0.3441\ \nu$	$1.6357\ \mu + 0.3355\ \nu$
	3	$1.9958\ \mu + 0.1160\ \nu$	$1.9641\ \mu + 0.1119\ \nu$
	4	$2.1118\ \mu + 0.0000\ \nu$	$2.0760\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.7138\ \nu$	$1.0000\ \mu + 0.7138\ \nu$
	2	$1.7138\ \mu + 0.4510\ \nu$	$1.6974\ \mu + 0.4410\ \nu$
	3	$2.1648\ \mu + 0.2310\ \nu$	$2.1245\ \mu + 0.2221\ \nu$
	4	$2.3958\ \mu + 0.0769\ \nu$	$2.3425\ \mu + 0.0743\ \nu$
	5	$2.4727\ \mu + 0.0000\ \nu$	$2.4169\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.7571\ \nu$	
	2	$1.7571\ \mu + 0.5292\ \nu$	
	3	$2.2864\ \mu + 0.3269\ \nu$	
	4	$2.6133\ \mu + 0.1649\ \nu$	
	5	$2.7782\ \mu + 0.0547\ \nu$	
	6	$2.8329\ \mu + 0.0000\ \nu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.7891\ \nu$	
	2	$1.7891\ \mu + 0.5885\ \nu$	
	3	$2.3776\ \mu + 0.4045\ \nu$	
	4	$2.7821\ \mu + 0.2463\ \nu$	
	5	$3.0283\ \mu + 0.1234\ \nu$	
	6	$3.1518\ \mu + 0.0409\ \nu$	
	7	$3.1927\ \mu + 0.0000\ \nu$	

TABLE 3.6b : Simple 3 Heuristic (Wedge Distribution)
(ratio of retrieval to optimal cost)

The asymptotic retrieval cost for a list of n elements with m buffers, whose retrieval probabilities are given by the Wedge distribution. The cost is in terms of the optimal number of buffer lookups and memory fetches.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.1102\nu$	$1.0000\mu + 1.1102\nu$
	2	$1.0367\mu + 1.1674\nu$	$1.0289\mu + 1.1326\nu$
	3	1.0498μ	1.0392μ
4	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.0862\nu$	$1.0000\mu + 1.0862\nu$
	2	$1.0323\mu + 1.1470\nu$	$1.0223\mu + 1.1183\nu$
	3	$1.0504\mu + 1.1600\nu$	$1.0337\mu + 1.1190\nu$
	4	1.0559μ	1.0380μ
5	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.0706\nu$	$1.0000\mu + 1.0706\nu$
	2	$1.0283\mu + 1.1275\nu$	$1.0184\mu + 1.1025\nu$
	3	$1.0475\mu + 1.1550\nu$	$1.0280\mu + 1.1105\nu$
	4	$1.0570\mu + 1.1529\nu$	$1.0334\mu + 1.1139\nu$
	5	1.0597μ	1.0358μ
6	0	1.0000ν	
	1	$1.0000\mu + 1.0599\nu$	
	2	$1.0250\mu + 1.1113\nu$	
	3	$1.0438\mu + 1.1442\nu$	
	4	$1.0554\mu + 1.1540\nu$	
	5	$1.0608\mu + 1.1492\nu$	
	6	1.0623μ	
7	0	1.0000ν	
	1	$1.0000\mu + 1.0521\nu$	
	2	$1.0223\mu + 1.0986\nu$	
	3	$1.0402\mu + 1.1327\nu$	
	4	$1.0527\mu + 1.1493\nu$	
	5	$1.0599\mu + 1.1522\nu$	
	6	$1.0633\mu + 1.1457\nu$	
	7	1.0309μ	

B. Batched K Heuristics with Limited Buffer Size

Here, requests are grouped into batch of k elements. The modified heuristic (MTF or TR) is applied only if the k requests in a batch are for the same record.

Modified Batched K MTF

The performance of this heuristic is studied in a manner analogous to that in the previous move to front heuristics. Each of the possible configurations of the buffer represents a state in the Markov chain which is described by the following equations :

i) $\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]$

$$= (1 - \sum_{j=2}^{n} p_{ij}^{k})\ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$+\ p_{i1}^{k}\ \{ \sum_{j=2}^{m} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}]$$

$$+ \sum_{x=m+1}^{n} \text{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \quad \}$$

ii) $\sum_{i} \text{Prob}[\pi_i] = 1$

THEOREM 3.3

Under the batched k heuristic with modified MTF rule, the stationary distribution of the state π_i is given by

$$\text{Prob}[\pi_i] = \text{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \prod_{j=1}^{m} \frac{p_{ij}^{k}}{\sum_{t=j}^{n} p_{it}^{k}}$$

PROOF OF THEOREM 3.3: See Appendix C.

In Appendix C, we show that

$$\mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im}) = \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

The above condition allows us to write :

$$\mu(m+1) = \mu(m) + \nu(m),\ 0 \le m \le n-1 \qquad \ldots\ldots(3.3)$$

as in Lemma 2.2. This condition in fact holds for all the move to front heuristics discussed so far.

Similarly, the retrieval cost of the batched k MTF heuristics is given in the same form as in the other move to front heuristics.

THEOREM 3.4

For a limited-buffer linear search system that adapts the modified batched k heuristics, the expected number of auxiliary memory fetches is given by

$$\nu(m) = \sum_{\pi_i \epsilon Q_m} [1 - \sum_{j=1}^{m} p_{ij}]\ \mathrm{Prob}[\pi_i], \qquad \text{and}$$

expected number of buffer lookups is given by

$$\mu(m) = 1 + \sum_{x=1}^{m-1} \sum_{\pi_i \epsilon Q_x} [1 - \sum_{j=1}^{x} p_{ij}]\ \mathrm{Prob}[\pi_i],$$

where Q_z is the set of all configurations of the buffer when the buffer size is z, and $s_i \epsilon Q_z$ is a particular state, $s_i = [R_{i1}R_{i2}\ldots R_{iz}]$. $\mathrm{Prob}[\pi_i]$ is given by Theorem 3.3.

PROOF:

Similar to the proof of Theorem 2.2.

The performance of this heuristic is studied in the same way as in the simple k heuristic. First of all, we shall examine the probability of record R_i in position j. Tables 3.7 and 3.8 below illustrate some values of $p_k(i,j)$, for various k's. It is interesting to note how the $p_k(i,j)$ changes with k, and also how it changes from simple k to the corresponding batched k heuristic. The values of $p_k(i,j)$ presented suggest that this batched k heuristic could perform more efficiently asymptotically for larger k. As k increases, each record R_i has a higher probability of being arranged in its respective optimal position, i. Furthermore, this batched k heuristic also indicates that it can be more efficient than the corresponding simple k heuristic. Numerical results for this move to front heuristic are directly calculated according to Theorem 3.4, and are presented in Tables 3.9a, 3.9b, 3.10a, 3.10b, 3.11a, 3.11b, 3.12a, and 3.12b. The results presented in the tables show that retrieval cost decreases as k increases, and the batched k heuristic can in fact be better than the corresponding simple k heuristic.

Modified Batched K TR

The Markov chain describing this batched k transposition heuristic is given as follows :

i) $\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]$

$$= (1 - \sum_{j=2}^{n} p_{ij}^{k})\ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$+ \sum_{j=1}^{m-1} p_{ij}^{k}\ \text{Prob}[R_{i1}\ldots R_{i(j-1)}R_{i(j+1)}R_{ij}\ldots R_{im}]$$

$$+ \ p_{im}^{k} \sum_{x=m+1}^{n} \text{Prob}[R_{i1}R_{i2}\ldots R_{im-1}R_{ix}]$$

ii) $\sum_{i} \text{Prob}[\pi_i] = 1$

Through the numerical examples presented in Tables 3.9 through 3.12 and the simulation results in Appendix D, it is indicated that the transposition rule can be more efficient than the corresponding move to front rule, in terms of both the expected number of buffer lookups, and the expected number of auxiliary memory fetches.

In this chapter, we have shown that, when a few extra bits of storage are utilized to remember a few previous requests, the asymtotic retrieval cost of the limited-buffer self-organizing scheme can be further reduced. From the numerical examples presented, the asymptotic retrieval cost of both the simple k and batched k heuristics decreases as k increases.

TABLE 3.7

The probability of record i in position j, assuming retrieval probabilities satisfy Zipf's distribution. Batched k move to front heuristic is adapted here.

record	probability of record i in position j					
i	1	2	3	4	5	6
n = 4						
	Simple MTF rule :					
1	0.4800	0.3085	0.1597	0.0518		
2	0.2400	0.3000	0.2800	0.1800		
3	0.1600	0.2200	0.3014	0.3186		
4	0.1200	0.1715	0.2589	0.4496		
	Batched 2 MTF rule :					
1	0.7024	0.2414	0.0512	0.0051		
2	0.1756	0.4375	0.2905	0.0964		
3	0.0780	0.2045	0.4046	0.3129		
4	0.0439	0.1167	0.2537	0.5857		
	Batched 3 MTF rule :					
1	0.8491	0.1398	0.0107	0.0003		
2	0.1061	0.6023	0.2484	0.0432		
3	0.0314	0.1812	0.5109	0.2765		
4	0.0133	0.0767	0.2300	0.6800		
n = 6						
	Simple MTF rule :					
1	0.4082	0.2814	0.1738	0.0913	0.0367	0.0086
2	0.2041	0.2291	0.2178	0.1777	0.1185	0.0527
3	0.1361	0.1662	0.1938	0.2026	0.1814	0.1199
4	0.1020	0.1291	0.1613	0.1952	0.2165	0.1958
5	0.0816	0.1053	0.1361	0.1760	0.2275	0.2734
6	0.0680	0.0888	0.1172	0.1571	0.2193	0.3495
	Batched 2 MTF rule:					
1	0.6705	0.2495	0.0665	0.0120	0.0013	0.0001
2	0.1676	0.3698	0.2732	0.1361	0.0455	0.0079
3	0.0745	0.1733	0.2824	0.2560	0.1577	0.0559
4	0.0419	0.0990	0.1768	0.2622	0.2602	0.1598
5	0.0268	0.0638	0.1177	0.1922	0.2946	0.3049
6	0.0186	0.0445	0.0833	0.1415	0.2407	0.4715
	Batched 3 MTF rule :					
1	0.8401	0.1457	0.0135	0.0007	0.0000	0.0000
2	0.1050	0.5578	0.2570	0.0689	0.0106	0.0007
3	0.0311	0.1679	0.3992	0.2730	0.1080	0.0208
4	0.0131	0.0711	0.1807	0.3431	0.2777	0.1143
5	0.0067	0.0364	0.0944	0.1964	0.3623	0.3037
6	0.0039	0.0211	0.0551	0.1180	0.2414	0.5605

TABLE 3.8

The probability of record i in position j, assuming retrieval probabilities satisfy the Wedge distribution.

record i	probability of record Ri in position j					
	1	2	3	4	5	6
n = 4						
	Simple MTF rule :					
1	0.4000	0.3159	0.2063	0.0778		
2	0.3000	0.3083	0.2619	0.1298		
3	0.2000	0.2413	0.3175	0.2413		
4	0.1000	0.1345	0.2143	0.5512		
	Batched 2 MTF rule :					
1	0.5333	0.3290	0.1231	0.0145		
2	0.3000	0.3994	0.2536	0.0470		
3	0.1333	0.2141	0.4794	0.1731		
4	0.0333	0.0575	0.1438	0.7654		
	Batched 3 MTF rule :					
1	0.6400	0.2988	0.0591	0.0020		
2	0.2700	0.5062	0.2100	0.0138		
3	0.0800	0.1726	0.6433	0.1040		
4	0.0100	0.0223	0.0875	0.8801		
n = 6						
	Simple MTF rule :					
1	0.2857	0.2485	0.2038	0.1496	0.0860	0.0264
2	0.2381	0.2279	0.2077	0.1719	0.1136	0.0408
3	0.1905	0.1970	0.2014	0.1929	0.1519	0.0662
4	0.1429	0.1576	0.1773	0.2034	0.2035	0.1153
5	0.0952	0.1109	0.1347	0.1757	0.2603	0.2231
6	0.0476	0.0581	0.0749	0.1065	0.1846	0.5282
	Batched 2 MTF rule :					
1	0.3956	0.3003	0.1926	0.0886	0.0214	0.0015
2	0.2747	0.2843	0.2419	0.1473	0.0474	0.0044
3	0.1758	0.2110	0.2625	0.2314	0.1057	0.0136
4	0.0989	0.1289	0.1858	0.3088	0.2319	0.0457
5	0.0440	0.0601	0.0927	0.1755	0.4564	0.1714
6	0.0110	0.0154	0.0245	0.0484	0.1373	0.7634
	Batched 3 MTF rule :					
1	0.4898	0.3190	0.1496	0.0381	0.0035	0.0001
2	0.2834	0.3446	0.2572	0.1006	0.0139	0.0003
3	0.1451	0.2092	0.3490	0.2399	0.0548	0.0020
4	0.0612	0.0947	0.1798	0.4464	0.2042	0.0137
5	0.0181	0.0289	0.0572	0.1548	0.6370	0.1039
6	0.0023	0.0036	0.0073	0.0201	0.0866	0.8800

TABLE 3.9a : Batched 2 Heuristic (Zipf's Distribution)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy Zipf's distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.5343\ \nu$	$1.0000\ \mu + 0.5343\ \nu$
	2	$1.5343\ \mu + 0.2209\ \nu$	$1.5095\ \mu + 0.2152\ \nu$
	3	$1.7552\ \mu + 0.0000\ \nu$	$1.7248\ \mu + 0.0000\ \nu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6029\ \nu$	$1.0000\ \mu + 0.6029\ \nu$
	2	$1.6029\ \mu + 0.3354\ \nu$	$1.5676\ \mu + 0.3250\ \nu$
	3	$1.9383\ \mu + 0.1459\ \nu$	$1.8799\ \mu + 0.1407\ \nu$
	4	$2.0842\ \mu + 0.0000\ \nu$	$2.0206\ \mu + 0.0000\ \nu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 0.6452\ \nu$	$1.0000\ \mu + 0.6029\ \nu$
	2	$1.6452\ \mu + 0.4062\ \nu$	$1.6047\ \mu + 0.3930\ \nu$
	3	$2.0515\ \mu + 0.2373\ \nu$	$1.9785\ \mu + 0.2271\ \nu$
	4	$2.2888\ \mu + 0.1064\ \nu$	$2.1972\ \mu + 0.1019\ \nu$
	5	$2.3952\ \mu + 0.0000\ \nu$	$2.2991\ \mu + 0.0000\ \nu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.6742\ \nu$	
	2	$1.6742\ \mu + 0.4550\ \nu$	
	3	$2.1292\ \mu + 0.3003\ \nu$	
	4	$2.4296\ \mu + 0.1808\ \nu$	
	5	$2.6104\ \mu + 0.0825\ \nu$	
	6	$2.6929\ \mu + 0.0000\ \nu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 0.6956\ \nu$	
	2	$1.6956\ \mu + 0.4909\ \nu$	
	3	$2.1865\ \mu + 0.3468\ \nu$	
	4	$2.5333\ \mu + 0.2357\ \nu$	
	5	$2.7691\ \mu + 0.1443\ \nu$	
	6	$2.9134\ \mu + 0.0667\ \nu$	
	7	$2.9801\ \mu + 0.0000\ \nu$	

TABLE 3.9b : Batched 2 Heuristic (Zipf's Distribution)
(ratio of the retrieval to optimal cost)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy Zipf's distribution. The cost is in terms of the optimal number of buffer lookups, and optimal number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	 Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1756\ \nu$	$1.0000\ \mu + 1.1756\ \nu$
	2	$1.0549\ \mu + 1.2151\ \nu$	$1.0378\ \mu + 1.1837\ \nu$
	3	$1.0726\ \mu$	$1.0540\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1594\ \nu$	$1.0000\ \mu + 1.1594\ \nu$
	2	$1.0545\ \mu + 1.1979\ \nu$	$1.0313\ \mu + 1.1607\ \nu$
	3	$1.0768\ \mu + 1.2158\ \nu$	$1.0444\ \mu + 1.1725\ \nu$
	4	$1.0855\ \mu$	$1.0524\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1480\ \nu$	$1.0000\ \mu + 1.0728\ \nu$
	2	$1.0533\ \mu + 1.1839\ \nu$	$1.0273\ \mu + 1.1454\ \nu$
	3	$1.0768\ \mu + 1.2040\ \nu$	$1.0385\ \mu + 1.1522\ \nu$
	4	$1.0888\ \mu + 1.2146\ \nu$	$1.0452\ \mu + 1.1632\ \nu$
	5	$1.0938\ \mu$	$1.0499\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.1392\ \nu$	
	2	$1.0518\ \mu + 1.1733\ \nu$	
	3	$1.0756\ \mu + 1.1931\ \nu$	
	4	$1.0889\ \mu + 1.2077\ \nu$	
	5	$1.0963\ \mu + 1.2132\ \nu$	
	6	$1.0996\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.1323\ \nu$	
	2	$1.0504\ \mu + 1.1647\ \nu$	
	3	$1.0740\ \mu + 1.1840\ \nu$	
	4	$1.0879\ \mu + 1.1995\ \nu$	
	5	$1.0965\ \mu + 1.2085\ \nu$	
	6	$1.1016\ \mu + 1.2105\ \nu$	
	7	$1.1039\ \mu$	

TABLE 3.10a : Batched 3 Heuristic (Zipf's Distribution)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy Zipf's distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	 Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.4955 \nu$	$1.0000 \mu + 0.4955 \nu$
	2	$1.4955 \mu + 0.2050 \nu$	$1.4822 \mu + 0.2028 \nu$
	3	$1.7004 \mu + 0.0000 \nu$	$1.6850 \mu + 0.0000 \nu$
4	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.5603 \nu$	$1.0000 \mu + 0.5603 \nu$
	2	$1.5603 \mu + 0.3105 \nu$	$1.5432 \mu + 0.3072 \nu$
	3	$1.8708 \mu + 0.1364 \nu$	$1.8431 \mu + 0.1337 \nu$
	4	$2.0071 \mu + 0.0000 \nu$	$1.9768 \mu + 0.0000 \nu$
5	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.6009 \nu$	$1.0000 \mu + 0.6009 \nu$
	2	$1.6009 \mu + 0.3760 \nu$	$1.5827 \mu + 0.3723 \nu$
	3	$1.9770 \mu + 0.2210 \nu$	$1.9442 \mu + 0.2163 \nu$
	4	$2.1980 \mu + 0.0998 \nu$	$2.1550 \mu + 0.0974 \nu$
	5	$2.2977 \mu + 0.0000 \nu$	$2.2524 \mu + 0.0000 \nu$
6	0	1.0000ν	
	1	$1.0000 \mu + 0.6293 \nu$	
	2	$1.6293 \mu + 0.4215 \nu$	
	3	$2.0507 \mu + 0.2793 \nu$	
	4	$2.3300 \mu + 0.1688 \nu$	
	5	$2.4988 \mu + 0.0776 \nu$	
	6	$2.5763 \mu + 0.0000 \nu$	
7	0	1.0000ν	
	1	$1.0000 \mu + 0.6504 \nu$	
	2	$1.6504 \mu + 0.4552 \nu$	
	3	$2.1056 \mu + 0.3224 \nu$	
	4	$2.4280 \mu + 0.2196 \nu$	
	5	$2.6476 \mu + 0.1350 \nu$	
	6	$2.7827 \mu + 0.0628 \nu$	
	7	$2.8455 \mu + 0.0000 \nu$	

TABLE 3.10b : Batched 3 Heuristic (Zipf's Distribution)
(ratio of the retrieval to optimal cost)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy Zipf's distribution. The cost is in terms of the optimal numebr of buffer lookups, and optimal number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost	
		Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.0902\ \nu$	$1.0000\ \mu + 1.0902\ \nu$
	2	$1.0282\ \mu + 1.1276\ \nu$	$1.0190\ \mu + 1.1155\ \nu$
	3	$1.0391\ \mu$	$1.0297\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.0775\ \nu$	$1.0000\ \mu + 1.0775\ \nu$
	2	$1.0265\ \mu + 1.1089\ \nu$	$1.0135\ \mu + 1.0971\ \nu$
	3	$1.0393\ \mu + 1.1367\ \nu$	$1.0239\ \mu + 1.1142\ \nu$
	4	$1.0454\ \mu$	$1.0296\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.0692\ \nu$	$1.0000\ \mu + 1.0692\ \nu$
	2	$1.0249\ \mu + 1.0959\ \nu$	$1.0133\ \mu + 1.0851\ \nu$
	3	$1.0377\ \mu + 1.1213\ \nu$	$1.0205\ \mu + 1.0974\ \nu$
	4	$1.0456\ \mu + 1.1393\ \nu$	$1.0251\ \mu + 1.1119\ \nu$
	5	$1.0493\ \mu$	$1.0281\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0634\ \nu$	
	2	$1.0236\ \mu + 1.0869\ \nu$	
	3	$1.0359\ \mu + 1.1097\ \nu$	
	4	$1.0442\ \mu + 1.1276\ \nu$	
	5	$1.0495\ \mu + 1.1412\ \nu$	
	6	$1.0520\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0588\ \nu$	
	2	$1.0224\ \mu + 1.0800\ \nu$	
	3	$1.0343\ \mu + 1.1007\ \nu$	
	4	$1.0426\ \mu + 1.1176\ \nu$	
	5	$1.0484\ \mu + 1.1307\ \nu$	
	6	$1.0522\ \mu + 1.1397\ \nu$	
	7	$1.0540\ \mu$	

TABLE 3.11a : Batched 2 Heuristic (Wedge Distribution)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy the Wedge distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	 Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.5714 \nu$	$1.0000 \mu + 0.5714 \nu$
	2	$1.5714 \mu + 0.2132 \nu$	$1.5536 \mu + 0.2042 \nu$
	3	$1.7846 \mu + 0.0000 \nu$	$1.7578 \mu + 0.0000 \nu$
4	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.6667 \nu$	$1.0000 \mu + 0.6667 \nu$
	2	$1.6667 \mu + 0.3667 \nu$	$1.6457 \mu + 0.3543 \nu$
	3	$2.0333 \mu + 0.1311 \nu$	$1.9856 \mu + 0.1221 \nu$
	4	$2.1644 \mu + 0.0000 \nu$	$2.1078 \mu + 0.0000 \nu$
5	0	1.0000ν	1.0000ν
	1	$1.0000 \mu + 0.7273 \nu$	$1.0000 \mu + 0.7273 \nu$
	2	$1.7273 \mu + 0.4736 \nu$	$1.7069 \mu + 0.4599 \nu$
	3	$2.2009 \mu + 0.2530 \nu$	$2.1445 \mu + 0.2369 \nu$
	4	$2.4539 \mu + 0.0882 \nu$	$2.3719 \mu + 0.0813 \nu$
	5	$2.5421 \mu + 0.0000 \nu$	$2.4532 \mu + 0.0000 \nu$
6	0	1.0000ν	
	1	$1.0000 \mu + 0.7692 \nu$	
	2	$1.7692 \mu + 0.5507 \nu$	
	3	$2.3199 \mu + 0.3515 \nu$	
	4	$2.6714 \mu + 0.1839 \nu$	
	5	$2.8554 \mu + 0.0633 \nu$	
	6	$2.9187 \mu + 0.0000 \nu$	
7	0	1.0000ν	
	1	$1.0000 \mu + 0.8000 \nu$	
	2	$1.8000 \mu + 0.6085 \nu$	
	3	$2.4085 \mu + 0.4295 \nu$	
	4	$2.8380 \mu + 0.2698 \nu$	
	5	$3.1078 \mu + 0.1392 \nu$	
	6	$3.2470 \mu + 0.0475 \nu$	
	7	$3.2946 \mu + 0.0000 \nu$	

TABLE 3.11b : Batched 2 Heuristic (Wedge Distribution)
(ratio of the retrieval to optimal cost)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy the Wedge distribution. The cost is in terms of the optimal number of buffer lookups, and optimal number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	Transposition rule
3	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1428\ \nu$	$1.0000\ \mu + 1.1428\ \nu$
	2	$1.0476\ \mu + 1.2789\ \nu$	$1.0357\ \mu + 1.2250\ \nu$
	3	$1.0707\ \mu$	$1.0547\ \mu$
4	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.1112\ \nu$	$1.0000\ \mu + 1.1117\ \nu$
	2	$1.0417\ \mu + 1.2223\ \nu$	$1.0286\ \mu + 1.1810\ \nu$
	3	$1.0700\ \mu + 1.3110\ \nu$	$1.0451\ \mu + 1.2210\ \nu$
	4	$1.0822\ \mu$	$1.0539\ \mu$
5	0	$1.0000\ \nu$	$1.0000\ \nu$
	1	$1.0000\ \mu + 1.0909\ \nu$	$1.0000\ \mu + 1.0909\ \nu$
	2	$1.0364\ \mu + 1.1840\ \nu$	$1.0241\ \mu + 1.1498\ \nu$
	3	$1.0649\ \mu + 1.2650\ \nu$	$1.0376\ \mu + 1.1845\ \nu$
	4	$1.0826\ \mu + 1.3223\ \nu$	$1.0464\ \mu + 1.2189\ \nu$
	5	$1.0895\ \mu$	$1.0514\ \mu$
6	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0769\ \nu$	
	2	$1.0320\ \mu + 1.1564\ \nu$	
	3	$1.0591\ \mu + 1.2303\ \nu$	
	4	$1.0788\ \mu + 1.2869\ \nu$	
	5	$1.0903\ \mu + 1.3298\ \nu$	
	6	$1.0945\ \mu$	
7	0	$1.0000\ \nu$	
	1	$1.0000\ \mu + 1.0667\ \nu$	
	2	$1.0286\ \mu + 1.1359\ \nu$	
	3	$1.0537\ \mu + 1.2027\ \nu$	
	4	$1.0738\ \mu + 1.2590\ \nu$	
	5	$1.0877\ \mu + 1.2997\ \nu$	
	6	$1.0954\ \mu + 1.3305\ \nu$	
	7	$1.0982\ \mu$	

TABLE 3.12a : Batched 3 Heuristic (Wedge Distribution)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy the Wedge distribution. The cost is in terms of μ, the expected number of buffer lookups, and ν, the expected number auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	Transposition rule
3	0	1.0000 ν	1.0000 ν
	1	1.0000 μ + 0.5463 ν	1.0000 μ + 0.5463 ν
	2	1.5463 μ + 0.1889 ν	1.5370 μ + 0.1846 ν
	3	1.7352 μ + 0.0000 ν	1.7215 μ + 0.0000 ν
4	0	1.0000 ν	1.0000 ν
	1	1.0000 μ + 0.6460 ν	1.0000 μ + 0.6460 ν
	2	1.6460 μ + 0.3378 ν	1.6316 μ + 0.3304 ν
	3	1.9838 μ + 0.1138 ν	1.9563 μ + 0.1106 ν
	4	2.0976 μ + 0.0000 ν	2.0669 μ + 0.0000 ν
5	0	1.0000 ν	1.0000 ν
	1	1.0000 μ + 0.7099 ν	1.0000 μ + 0.7099 ν
	2	1.7099 μ + 0.4456 ν	1.6944 μ + 0.4365 ν
	3	2.1555 μ + 0.2274 ν	2.1188 μ + 0.2199 ν
	4	2.3829 μ + 0.0759 ν	2.3352 μ + 0.0738 ν
	5	2.4588 μ + 0.0000 ν	2.4090 μ + 0.0000 ν
6	0	1.0000 ν	
	1	1.0000 μ + 0.7543 ν	
	2	1.7543 μ + 0.5249 ν	
	3	2.2792 μ + 0.3229 ν	
	4	2.6021 μ + 0.1629 ν	
	5	2.7651 μ + 0.0542 ν	
	6	2.8193 μ + 0.0000 ν	
7	0	1.0000 ν	
	1	1.0000 μ + 0.7870 ν	
	2	1.7870 μ + 0.5850 ν	
	3	2.3720 μ + 0.4007 ν	
	4	2.7727 μ + 0.2436 ν	
	5	3.0163 μ + 0.1223 ν	
	6	3.1386 μ + 0.0407 ν	
	7	3.1793 μ	

TABLE 3.12b : Batched 3 Heuristic (Wedge Distribution)
(ratio of the retrieval to optimal cost)

The asymptotic retrieval cost for the list of n elements with m buffers, whose retrieval probabilities satisfy the Wedge distribution. The cost is in terms of the optimal number of buffer lookups, and optimal number of auxiliary memory fetches.

number of elements	buffer size	asymptotic retrieval cost Move To Front rule	Transposition rule
3	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.0926\nu$	$1.0000\mu + 1.0926\nu$
	2	$1.0309\mu + 1.1332\nu$	$1.0247\mu + 1.1074\nu$
	3	1.0411μ	1.0329μ
4	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.0767\nu$	$1.0000\mu + 1.0767\nu$
	2	$1.0288\mu + 1.1260\nu$	$1.0198\mu + 1.1013\nu$
	3	$1.0441\mu + 1.1380\nu$	$1.0296\mu + 1.1060\nu$
	4	1.0488μ	1.0335μ
5	0	1.0000ν	1.0000ν
	1	$1.0000\mu + 1.0648\nu$	$1.0000\mu + 1.0634\nu$
	2	$1.0259\mu + 1.1140\nu$	$1.0166\mu + 1.0913\nu$
	3	$1.0430\mu + 1.1370\nu$	$1.0252\mu + 1.0995\nu$
	4	$1.0513\mu + 1.1379\nu$	$1.0302\mu + 1.1064\nu$
	5	1.0538μ	1.0324μ
6	0	1.0000ν	
	1	$1.0000\mu + 1.0560\nu$	
	2	$1.0233\mu + 1.1023\nu$	
	3	$1.0405\mu + 1.1302\nu$	
	4	$1.0508\mu + 1.1400\nu$	
	5	$1.0558\mu + 1.1387\nu$	
	6	$1.0572\mu + 1.0000\nu$	
7	0	1.0000ν	
	1	$1.0000\mu + 1.0493\nu$	
	2	$1.0211\mu + 1.0920\nu$	
	3	$1.0378\mu + 1.1221\nu$	
	4	$1.0491\mu + 1.1367\nu$	
	5	$1.0557\mu + 1.1419\nu$	
	6	$1.0588\mu + 1.1401\nu$	
	7	1.0598μ	

IV. Conclusion

We have shown how self-organizing heuristics can be adapted to a limited buffer size linear search system to improve performance. Through numerical examples and simulation results, we have shown that the transposition heuristics are generally better than the corresponding move to front heuristics. Furthermore, the study of k in a row heuristics exihibits that better asymptotic performance can be attained at the expense of a few extra bits of storage.

A related application of the limited buffer size self-organizing scheme is in the construction of paging algorithms for memory management. After every page reference, the list of pages currently residing in the main memory can be reorganized. In case of a page fault, a page will be removed to make room for the new page, according to one of the limited-buffer self-organizing heuristics discussed so far. The "Least Recently Used" (LRU) paging algorithm corresponds to the modified simple move to front heuristic. Similarly, the transposition rule or "k in a row" heuristics could be incorporated into the paging algorithms to improve performance.

So far, our study on the transposition heuristic for the limited-buffer linear search system has been limited to the study of numerical examples and simulation results, as the actual retrieval cost is difficult to derive. It is hoped that in the future, the class of transposition

heuristics can be treated to more analytical study, so that more can be said of the performance in general.

As for the conventional self-organizing heuristics, the convergence rate of the limited-buffer self-organizing heuristics cannot be overlooked. We have seen that, in the conventional system, although the transposition rule and the "k in a row" heuristics are asymptotically more efficient, their convergence rates are generally slower than the simple move to front heuristics. Therefore, the simple move to front heuristic may be a more favorable choice when there are fewer requests in the linear search system or when the access frequency changes from time to time. For this limited-buffer scheme, we may expect similarly slower convergence to the asymptotic efficiency by the asymptotically more efficient transposition and the "k in a row" heuristics. Here, there are two performance measures of interest, namely expected number of buffer lookups, and the expected number of auxiliary memory fetches. The study of the relative convergence rate of the various heuristics remains an open problem.

One major assumption of our study is the independence assumption, i.e., the retrieval probability of each record remains unchanged and independent at all times. When there is a high correlation between successive requests, the class of move to front heuristics may outperform the corresponding transposition rule. Lam, Leung, and Siu[11] examine the

simple move to front heuristics when each access is assumed to be dependent on the access immediately preceding it. However, the study of transposition rule with dependent accesses still remains an open problem.

Bibliography

[1] Bitner, J.R., "Heuristics that Dynamically Organize Data Structures," *SIAM J. Cmput.*, *8* (1979), pp. 82-100.

[2] Burville, P.J. and Kingman, J.F.C., "On a Model for Storage and Search," *J. Appl. Probability*, *10* (1973), pp. 697-701.

[3] Coffman, E.G. and Denning, P.J., *Operating Systems Theory*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

[4] Franasek, P.A. and Wagner, T.J., "Some Distribution-Free Aspects of Paging Algorithm Performance," *JACM*, *21* (1974), pp. 31-39.

[5] Gonnet, G.H., Munro, J.I. and Suwanda, H., "Exegesis of Self-Organizing Linear Search," *SIAM J. Cmput.*, *10* (1981), pp.613-637.

[6] Hendricks, W.J., "The Stationary Distribution of an Interesting Markov Chain," *J. Appl. Probability*, *9* (1972), pp. 231-233.

[7] Hendricks, W.J., "An Extension of a Theorem Concerning

an Interesting Markov Chain," *J. Appl. Probability, 10* (1973), pp. 886-890.

[8] Hendricks, W.J., "An Account of Self-Organizing Systems," *SIAM J. Cmput., 5* (1976), pp. 715-723.

[9] Kemeny, J.G. and Snell, J.L., *Finite Markov Chains*, Van Nostrand, Princeton, 1960.

[10] Knuth, D.E., *The Art of Computer Programming, Vol III*, Searching and Sorting, Addison-Wesley, Don Mills, Ont., 1973.

[11] Lam, K., Leung, M.Y. and Siu M.K., "Self-Organizing Files with Dependent Accesses," University of Hong Kong, 1982.

[12] McCabe, J., "On Serial Files with Relocatable Records", *Operations Res., 12* (1965), pp. 609-618.

[13] Rivest, R., "On Self-Organizing Sequential Search Heuristics," *Comm. ACM, 19* (1976), pp. 63-67.

[14] Tanenbaum, A., "Simulations of Dynamic Sequential Search Algorithms," *Comm. ACM, 21* (1978), pp. 790-791.

Appendix A. Expected Number of Auxiliary Memory Fetches

The expected number of auxiliary memory fetches can be shown to be as follow :

$$\nu(m) = \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \{ (p_i - p_{m+1}) [1 - \sum_{j=1}^{m} p(i,j)] \}$$

$$+ \sum_{i=m+2}^{n} \{ (p_{m+1} - p_i) \sum_{j=1}^{m} p(i,j) \} \qquad \text{.....(A.1)}$$

PROOF :

Given n records and a buffer size of m, the expected number of memory fetches is defined to be :

$\nu(m) = \sum_{i=1}^{n} \{ p_i [1 - \sum_{j=1}^{m} p(i,j)] \}$, where $p(i,j)$ denotes the probability that record R_i is in position j

Therefore,

$\nu(m)$

$$= \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \{ p_i [1 - \sum_{j=1}^{m} p(i,j)] \} - \sum_{i=m+1}^{n} \{ p_i \sum_{j=1}^{m} p(i,j)\}$$

$$= \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \{ p_i [1 - \sum_{j=1}^{m} p(i,j)] \} - \sum_{i=m+2}^{n} \{ p_i \sum_{j=1}^{m} p(i,j)\}$$

$$- p_{m+1} \sum_{j=1}^{m} p(m+1,j) \qquad \text{.....(A.2)}$$

Before proceeding to simplify the above expression, we shall first show that

$$\sum_{j=1}^{m} p(m+1,j) = \{ \sum_{i=1}^{m} [1 - \sum_{j=1}^{m} p(i,j)] \} - \{ \sum_{i=m+2}^{n} \sum_{j=1}^{m} p(i,j) \} \qquad \text{.....(A.3)}$$

R.H.S of (A.3)

$$= \left\{ \sum_{j=1}^{m} \left[1 - \sum_{i=1}^{m} p(i,j) \right] \right\} - \left\{ \sum_{i=m+2}^{n} \sum_{j=1}^{m} p(i,j) \right\}$$

$$= \left\{ \sum_{j=1}^{m} \left[\sum_{i=m+1}^{n} p(i,j) \right] \right\} - \left\{ \sum_{i=m+2}^{n} \sum_{j=1}^{m} p(i,j) \right\}$$

$$= \left\{ \sum_{i=m+1}^{n} \left[\sum_{j=1}^{m} p(i,j) \right] \right\} - \left\{ \sum_{i=m+2}^{n} \sum_{j=1}^{m} p(i,j) \right\}$$

$$= \sum_{j=1}^{m} p(m+1,j) = \text{L.H.S. of (A.3)}$$

Therefore, the expression (A.2) can be further reduced to

$\nu(m)$

$$= \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \left\{ p_i \left[1 - \sum_{j=1}^{m} p(i,j) \right] \right\} - \sum_{i=m+2}^{n} \left\{ p_i \sum_{j=1}^{m} p(i,j) \right\}$$

$$- p_{m+1} \left\{ \sum_{i=1}^{m} \left[1 - \sum_{j=1}^{m} p(i,j) \right] \right\} + p_{m+1} \left\{ \sum_{i=m+2}^{n} \sum_{j=1}^{m} p(i,j) \right\}$$

$$= \sum_{i=m+1}^{n} p_i + \sum_{i=1}^{m} \left\{ (p_i - p_{m+1}) \left[1 - \sum_{j=1}^{m} p(i,j) \right] \right\}$$

$$+ \sum_{i=m+2}^{n} \left\{ (p_{m+1} - p_i) \sum_{j=1}^{m} p(i,j) \right\} \qquad \blacksquare$$

Appendix B. On the Markov Chain for the Modified Simple K MTF Heuristic

In Chapter 3, we have shown that the limited buffer linear search system can be modelled as a Markov Chain, whereby the state is described by each of the possible configurations of the buffer. Let π_i be one of the ${}^nP_m$ possible states of the m-size buffer, $\pi_i \in Q_m$. Let $\pi_i = [R_{i1}R_{i2}\ldots R_{im}]$, and let those records that are not in the buffer for this configuration be labelled as $R_{i(m+1)}$ through R_n.

Let $(R_{i1}R_{i2}\ldots R_{im})_0$ be the initial state where records $R_{i1}, R_{i2}, \ldots, R_n$ are arranged in this order, and let $(R_{i1}R_{i2}\ldots R_{im})_{y(z)}$, $1 \le y \le n$ and $1 \le z \le k-1$ be the state where the z previous requests are all for record R_y. It is shown in Chapter 3 that the Markov Chain must satisfy the following equations :

B1) $\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0$

$$= p_{i1} \{ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{i1(k-1)} + \sum_{j=2}^{m} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}]_{i1(k-1)} + \sum_{x=m+1}^{n} \text{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}]_{i1(k-1)} \} \qquad \ldots\ldots\ldots\ldots(B1.1)$$

where

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)}$$

$$= p_j \ \{ \ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{\substack{y=1 \\ y\neq j}}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)}] \ \} \qquad \ldots\ldots\ldots\ldots(B1.2)$$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} = p_j \ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t-1)}, \quad 2 \leq t \leq k-1 \qquad \ldots\ldots\ldots\ldots(B1.3)$$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0 + \sum_{y=1}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)} \qquad \ldots\ldots\ldots\ldots(B1.4)$$

B2) $\sum_{i} \mathrm{Prob}[\pi_i] = 1$

For ease of notation, we shall denote $\sum_{t=1}^{k-1} p_j^{\,t}$ by X_j for the discussion that follows.

THEOREM B.1:

Under the modified simple k MTF heuristic, the stationary distribution of the state π_i is given by

$$\mathrm{Prob}[\pi_i] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \prod_{j=1}^{m} \frac{\left[\ p_{ij}^{\,k} \prod_{t=j+1}^{n} (1+X_{it}) \ \right]}{\left[\ \sum_{x=j}^{n} \{ \ p_{ix}^{\,k} + \prod_{\substack{t=j \\ t\neq x}}^{n} (1+X_{it}) \ \} \ \right]}$$

or it can be written as

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}^{k}\right]\left[\prod_{j=2}^{m}(1+X_{ij})^{j-1}\right]\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m}\right]}{\prod_{j=1}^{m}\left[\sum_{x=j}^{n}\left\{p_{ix}^{k}+\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

The proof will be given later. First of all, we shall present the following lemmas.

Define

$$\phi'_{z} = \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m}(1+X_{ij})^{j-2}\right]\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m-1}\right](1+X_{i1})^{z-1}}{\prod_{j=2}^{z}\left[p_{i1}^{k}\prod_{t=j}^{n}(1+X_{it})+\sum_{x=j}^{n}\left\{p_{ix}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]} *$$

$$\frac{1}{\left[\prod_{j=z+1}^{m}\left\{\sum_{x=j}^{n}\left[p_{ix}^{k}\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right]\right\}\right]}, \quad 2 \leq z \leq m-1$$

and,

$$\phi'_{m} = \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m}(1+X_{ij})^{j-2}\right]\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m-1}\right](1+X_{i1})^{m-1}}{\prod_{j=2}^{m}\left[p_{i1}^{k}\prod_{t=j}^{n}(1+X_{it})+\sum_{x=j}^{n}\left\{p_{ix}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

LEMMA B.1

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}] \quad + \quad \phi'_{y+1} \quad = \phi'_{y}$$

$$2 \le y \le m-1$$

PROOF:

Case 1 : $2 \le y \le m-2$

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}]$$

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}{}^{k}\right]\left[\prod_{j=3}^{y} (1+X_{ij})^{j-2}\right](1+X_{i1})^{y-1}\left[\prod_{j=y+1}^{m} (1+X_{ij})^{j-1}\right]}{\prod_{j=2}^{y+1}\left[\ p_{i1}{}^{k}\prod_{t=j}^{n}(1+X_{it}) + \sum_{x=j}^{n}\{\ p_{ix}{}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\ne x}}^{n}(1+X_{it})\ \}\ \right]} \ast$$

$$\frac{\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m}\ \right]}{\left[\prod_{j=y+1}^{m}\{\ \sum_{x=j}^{n}\ [\ p_{ix}{}^{k}\ \prod_{\substack{t=j\\ t\ne x}}^{n}(1+X_{it})\ ]\ \}\ \right]}$$

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}] \quad + \quad \phi'_{y+1}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}{}^{k}\right]\left[\prod_{j=3}^{m} (1+X_{ij})^{j-2}\right]\left[\prod_{j=m+1}^{n} (1+X_{ij})^{m-1}\right]\ (1+X_{i1})^{y-1}}{\prod_{j=2}^{y+1}\left[\ p_{i1}{}^{k}\prod_{t=j}^{n}(1+X_{it}) + \sum_{x=j}^{n}\{\ p_{ix}{}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\ne x}}^{n}(1+X_{it})\ \}\ \right]} \ast$$

$$\frac{\left[p_{i1}{}^{k}\ \prod_{j=y+1}^{n}(1+X_{ij})\right] + (1+X_{i1})\left[\sum_{x=y+1}^{n}\{\ p_{ix}{}^{k}\prod_{\substack{t=j\\ t\ne x}}^{n}(1+X_{it})\ \}\right]}{\left[\prod_{j=y+1}^{m}\{\ \sum_{x=j}^{n}\ [\ p_{ix}{}^{k}\ \prod_{\substack{t=j\\ t\ne x}}^{n}(1+X_{it})\ ]\ \}\ \right]}$$

$$= \phi'_{y}$$

Case 2 : y = m-1

$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{i(m-1)}R_{i1}R_{im}] + \phi'_m$

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m-1}(1+X_{ij})^{j-2}\right](1+X_{i1})^{m-2}(1+X_{im})^{m-1}\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m}\right]}{\prod_{j=2}^{m}\left[p_{i1}^{k}\prod_{t=j}^{n}(1+X_{it}) + \sum_{x=j}^{n}\left\{p_{ix}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

$$* \frac{1}{\left[\sum_{x=m}^{n}\left\{p_{ix}^{k}\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

$$+ \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m}(1+X_{ij})^{j-2}\right]\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m-1}\right](1+X_{i1})^{m-1}}{\prod_{j=2}^{m}\left[p_{i1}^{k}\prod_{t=j}^{n}(1+X_{it}) + \sum_{x=j}^{n}\left\{p_{ix}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m}(1+X_{ij})^{j-2}\right]\left[\prod_{j=m+1}^{n}(1+X_{ij})^{m-1}\right](1+X_{i1})^{m-2}}{\prod_{j=2}^{m}\left[p_{i1}^{k}\prod_{t=j}^{n}(1+X_{it}) + \sum_{x=j}^{n}\left\{p_{ix}^{k}(1+X_{i1})\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]} *$$

$$\frac{p_{i1}^{k}\prod_{t=m}^{n}(1+X_{it}) + (1+X_{i1})\sum_{x=m}^{n}\left\{p_{ix}^{k}\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}}{\left[\sum_{x=m}^{n}\left\{p_{ix}^{k}\prod_{\substack{t=j\\ t\neq x}}^{n}(1+X_{it})\right\}\right]}$$

$= \phi'_{m-1}$ ■

LEMMA B.2

$$\mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im}) = \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

PROOF:

This lemma follows in a manner analogous to Lemma 2.1b. The analytical proof is given as follow :

$\mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}^{\,k}\right] p_{ix}^{\,k} \left[\prod_{j=2}^{m} (1+X_{ij})^{j-1}\right] (1+X_{ix})^{m} \left[\prod_{\substack{j=m+1\\ j\neq x}}^{n} (1+X_{ij})^{m+1}\right]}{\prod_{j=1}^{m+1} \left[\sum_{y=j}^{n} \left\{ p_{iy}^{\,k} \prod_{\substack{t=j\\ t\neq y}}^{n} (1+X_{it}) \right\}\right]}$$

$$\therefore \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

$$= \frac{\left[\prod_{j=1}^{m} p_{ij}^{\,k}\right] \left[\prod_{j=2}^{m} (1+X_{ij})^{j-1}\right] \left[\prod_{j=m+1}^{n} (1+X_{ij})^{m}\right]}{\prod_{j=1}^{m} \left[\sum_{y=j}^{n} \left\{ p_{iy}^{\,k} \prod_{\substack{t=j\\ t\neq y}}^{n} (1+X_{it}) \right\}\right]} \quad * \quad \frac{\sum_{x=m+1}^{n} \left[p_{ix}^{\,k} \prod_{\substack{j=m+1\\ j\neq x}}^{n} (1+X_{ij}) \right]}{\sum_{y=m+1}^{n} \left[p_{iy}^{\,k} \prod_{\substack{j=m+1\\ j\neq y}}^{n} (1+X_{ij}) \right]}$$

$$= \mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im}) \qquad \blacksquare$$

PROOF OF THEOREM B.1:

The stationary distribution of the state $[\pi_i]$ must satisfy the two conditions, (B1) and (B2), as stated above.

Case 1 : to show that (B1) is satisfied :

First, we shall express $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_o$ and $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)}$ in terms of $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$.

From (B1.2), (B1.3) and (B1.4) we obtain

$$\begin{aligned}
&\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} \\
&\quad = p_j^{\,t-1}\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)} \qquad \ldots\ldots\ldots\ldots(B.5) \\
&\quad = p_j^{\,t}\ \Big\{\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_o \\
&\qquad\qquad + \sum_{\substack{y=1\\ y\neq j}}^{n}\ \sum_{z=1}^{k-1}\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)}\ \Big\} \\
&\quad = p_j^{\,t}\ \Big\{\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] - \sum_{z=1}^{k-1}\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(z)}\ \Big\}
\end{aligned}$$

$$\begin{aligned}
\therefore\ \ &\sum_{t=1}^{k-1}\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} \\
&\qquad = \Big(\ \sum_{t=1}^{k-1}\ p_j^{\,t}\ \Big)\ \Big\{\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] \\
&\qquad\qquad\qquad - \sum_{z=1}^{k-1}\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(z)}\ \Big\}
\end{aligned}$$

$$\begin{aligned}
\Rightarrow\ \ &\sum_{t=1}^{k-1}\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} \\
&\qquad = \frac{\displaystyle\sum_{t=1}^{k-1}\ p_j^{\,t}}{1+\displaystyle\sum_{t=1}^{k-1}\ p_j^{\,t}}\ \ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] \qquad \ldots\ldots\ldots\ldots(B.6)
\end{aligned}$$

From (B.5) we get $\sum_{t=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)}$

$$= \left(\sum_{t=1}^{k-1} p_j^{\,t-1} \right) \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)}$$

Comparing with (B.6),

$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(1)}$

$$= \frac{\sum_{t=1}^{k-1} p_j^{\,t}}{\left[\sum_{t=1}^{k-1} p_j^{\,t-1} \right]\left[1 + \sum_{t=1}^{k-1} p_j^{\,t} \right]} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \frac{p_j}{1 + \sum_{t=1}^{k-1} p_j^{\,t}} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$\therefore \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} = \frac{p_j^{\,t}}{1 + \sum_{t=1}^{k-1} p_j^{\,t}} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

or, using the notation of $X_j = \sum_{t=1}^{k-1} p_j^{\,t}$

$$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{j(t)} = \frac{p_j^{\,t}}{(1+X_j)} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0$

$$= \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] - \sum_{y=1}^{n} \sum_{z=1}^{k-1} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{y(z)} \quad \text{by (B1.4)}$$

$$= \left[1 - \sum_{y=1}^{n} \frac{X_y}{(1+X_y)} \right] \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] \quad \text{by (B.6)}$$

(B1.1) can then be rewritten as

$$\text{Prob}[R_{i1}R_{i2}\ldots R_{im}]_0$$

$$= \frac{p_{i1}^{\,k}}{(1+X_{i1})} \Big\{ \text{Prob}[R_{i1}R_{i2}\ldots R_{im}] + \sum_{j=2}^{m} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \sum_{x=m+1}^{n} \text{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \Big\}$$

$$\ldots\ldots\ldots\ldots(\text{B1.1a})$$

Substitute prob[π_i] of Theorem B.1 into

$$\sum_{j=2}^{m} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \sum_{x=m+1}^{n} \text{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \qquad \text{gives}$$

$$\sum_{j=2}^{m-1} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] + \text{Prob}[R_{i2}\ldots R_{im}R_{i1}]$$

$$+ \sum_{x=m+1}^{n} \Big[p_{ix}^{\,k} \;\Big\{ (1+X_{i1}) \prod_{\substack{j=m+1 \\ j \neq x}}^{n} (1+X_{ij}) \Big\} \Big] *$$

$$\frac{\Big[\prod_{j=2}^{m} p_{ij}^{\,k}\Big]\Big[\prod_{j=3}^{m} (1+X_{ij})^{j-2}\Big](1+X_{i1})^{m-1}\Big[\prod_{j=m+1}^{n} (1+X_{ij})^{m-1}\Big]}{\prod_{j=2}^{m+1} \Big[p_{i1}^{\,k} \prod_{t=j}^{n} (1+X_{it}) + \sum_{x=j}^{n} \Big\{ p_{ix}^{\,k}\, (1+X_{i1}) \prod_{\substack{t=j \\ t \neq x}}^{n} (1+X_{it}) \Big\} \Big]}$$

$$= \sum_{j=2}^{m-1} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}] \quad +$$

$$\Big(\Big[p_{i1}^{\,k} \prod_{j=m+1}^{n} (1+X_{ij})\Big] + \sum_{x=m+1}^{n} \Big[p_{ix}^{\,k} \;\Big\{ (1+X_{i1}) \prod_{\substack{j=m+1 \\ j \neq x}}^{n} (1+X_{ij}) \Big\} \Big]\Big) *$$

$$\frac{\Big[\prod_{j=2}^{m} p_{ij}^{\,k}\Big]\Big[\prod_{j=3}^{m} (1+X_{ij})^{j-2}\Big](1+X_{i1})^{m-1}\Big[\prod_{j=m+1}^{n} (1+X_{ij})^{m-1}\Big]}{\prod_{j=2}^{m+1} \Big[p_{i1}^{\,k} \prod_{t=j}^{n} (1+X_{it}) + \sum_{x=j}^{n} \Big\{ p_{ix}^{\,k}\, (1+X_{i1}) \prod_{\substack{t=j \\ t \neq x}}^{n} (1+X_{it}) \Big\} \Big]}$$

$$= \sum_{j=2}^{m-1} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}] \quad + $$

$$\frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m} (1+X_{ij})^{j-2}\right](1+X_{i1})^{m-1}\left[\prod_{j=m+1}^{n} (1+X_{ij})^{m-1}\right]}{\prod_{j=2}^{m} \left[p_{i1}^{k} \prod_{t=j}^{n} (1+X_{it}) + \sum_{x=j}^{n} \{ p_{ix}^{k} \ (1+X_{i1}) \prod_{\substack{t=j \\ t \neq x}}^{n} (1+X_{it}) \} \right]}$$

$$= \sum_{j=2}^{m-1} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}] \quad + \quad \phi'_{m}$$

$$= \sum_{j=2}^{m-2} \text{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}] \quad + \quad \phi'_{m-1}$$

(by applying Lemma B.1)

$= \phi'_{2}$ (by applying Lemma B.1 m-3 times)

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[\prod_{j=3}^{m} (1+X_{ij})^{j-2}\right](1+X_{i1})\left[\prod_{j=m+1}^{n} (1+X_{ij})^{m-1}\right]}{\left[p_{i1}^{k} \prod_{t=2}^{n} (1+X_{it}) + \sum_{x=2}^{n} \{ p_{ix}^{k} \ (1+X_{i1}) \prod_{\substack{t=2 \\ t \neq y}}^{n} (1+X_{it}) \} \right]} \quad *$$

$$\frac{1}{\left[\prod_{j=3}^{m} \{ \sum_{x=j}^{n} [p_{ix}^{k} \prod_{\substack{t=j \\ t \neq x}}^{n} (1+X_{it})] \} \right]}$$

$$= \frac{\left[\sum_{x=2}^{n} \{ p_{ix}^{k} \prod_{\substack{t=2 \\ t \neq x}}^{n} (1+X_{it}) \} \right] (1+X_{i1})}{(p_{i1}^{k}) \left[\prod_{j=2}^{n} (1+X_{ij}) \right]} \ \text{Prob}[R_{i1}R_{i2}...R_{im}]$$

$\therefore$ R.H.S. of (B1.1a) becomes

$$\frac{p_{i1}{}^{k}}{(1+X_{i1})}\{\,\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] + \phi'_{2}\,\}$$

$$= \left\{ \frac{p_{i1}{}^{k}}{1 + X_{i1}} + \frac{\left[\sum\limits_{x=2}^{n} \{p_{ix}{}^{k} \prod\limits_{\substack{t=2\\ t\neq x}}^{n} (1+X_{it})\}\right]}{\left[\prod\limits_{j=2}^{n} (1+X_{ij})\right]} \right\} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \left[\sum_{t=1}^{n} \frac{p_{it}{}^{k}}{1 + X_{it}}\right] \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \left[1 - \sum_{t=1}^{n} p_{it} + \sum_{t=1}^{n} \frac{p_{it}{}^{k}}{1 + X_{it}}\right] \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \left[1 - \sum_{t=1}^{n} \frac{p_{it}(1+X_{it}) - p_{it}{}^{k}}{1 + X_{it}}\right] \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \left[1 - \sum_{t=1}^{n} \frac{X_{it}}{1 + X_{it}}\right] \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]_{0}$$

= R.H.S of (B1.1a)

Case 2 : to show that (B2) is satisfied.

Lemma B.2 allows us to write down :

$$\Sigma\ \mathrm{Prob}_m(R_i \ldots) = \mathrm{Prob}_1(R_i), \quad 1 \le m \le n$$

as in Lemma 2.2.

$$\Sigma\ \mathrm{Prob}[\pi \text{ with } R_i \text{ in position } 1]$$

$$= \mathrm{Prob}^1(R_i)$$

$$= \frac{p_i{}^k \prod_{\substack{t=1 \\ t \neq i}}^{n} (1+X_t)}{\sum_{j=1}^{n} \left[p_j{}^k \prod_{\substack{t=1 \\ t \neq j}}^{n} (1+X_t) \right]}$$

Hence,

$$\sum_i \pi_i = \sum_{i=1}^{n} \{ \Sigma\ \mathrm{Prob}[\pi \text{ with } R_i \text{ in position } 1] \}$$

$$= \frac{\sum_{i=1}^{n} \left[p_i{}^k \prod_{\substack{t=1 \\ t \neq i}}^{n} (1+X_t) \right]}{\sum_{j=1}^{n} \left[p_j{}^k \prod_{\substack{t=1 \\ t \neq j}}^{n} (1+X_t) \right]}$$

$$= 1 \quad \blacksquare$$

Appendix C. On the Markov Chain for the Modified Batched K Heuristic

Let $\pi_i = [R_{i1}R_{i2}\ldots R_{im}]$ be a particular state of the Markov chain, where each state represents one particular configuration of the buffer, under the modified batched k heuristic. Let the records that do not reside in buffer for this particular state π_i be denoted by R_{m+1} through R_n.

In Chapter 3, we have shown that the Markov Chain must satisfy the following equations :

C1) $\mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$

$$= (1 - \sum_{j=2}^{n} p_{ij}^{k})\ \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$+ \ p_{i1}^{k} \ \{ \sum_{j=2}^{m} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}]$$

$$+ \sum_{x=m+1}^{n} \mathrm{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \qquad \}$$

C2) $\sum_{i} \mathrm{Prob}[\pi_i] = 1$

THEOREM C.1

Under the batched k heuristic with modified MTF rule, the stationary distribution of the state π_i is given by

$$\mathrm{Prob}[\pi_i] = \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}] = \prod_{j=1}^{m} \frac{p_{ij}^{k}}{\sum_{t=j}^{n} p_{it}^{k}}$$

We shall study the two lemmas below before proceeding to the proof.

Define

$$\phi^{+}_{z} = \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[\prod_{j=2}^{z} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]\left[\prod_{j=z+1}^{m} (\sum_{t=j}^{n} p_{it}^{k})\right]}, \quad 2 \le z \le m-1$$

and,

$$\phi^{+}_{m} = \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[\prod_{j=2}^{m} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]}$$

<u>LEMMA C.1</u>

$$\mathrm{Prob}[R_{i2}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}] + \phi^{+}_{y+1} = \phi^{+}_{y}, \quad 2 \le y \le m-1$$

<u>PROOF:</u>

Case 1 : 2 ≤ y ≤ m-2

$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}]$

$$= \frac{\prod_{j=1}^{m} p_{ij}^{k}}{\left[\prod_{j=2}^{y+1} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k}\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it}^{k})\right]}$$

$$\therefore \quad \mathrm{Prob}[R_{i2}R_{i3}\ldots R_{iy}R_{i1}R_{i(y+1)}\ldots R_{im}] + \phi^{+}_{y+1}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[p_{i1}^{k} + \sum_{t=y+1}^{n} p_{it}^{k}\right]}{\left[\prod_{j=2}^{y+1} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it}^{k})\right]}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]}{\left[\prod_{j=2}^{y} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]\left[\prod_{j=y+1}^{m} (\sum_{t=j}^{n} p_{it}^{k})\right]}$$

$$= \phi^{+}_{y}$$

Case 2 : y = m-1

$$\mathrm{Prob}[R_{i2}R_{i3}\ldots R_{i(m-1)}R_{i1}R_{im}] + \phi^{+}_{m}$$

$$= \frac{\prod_{j=1}^{m} p_{ij}^{k}}{\left[\prod_{j=2}^{m}(p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]\left[\sum_{t=m}^{n} p_{it}^{k}\right]} + \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[\prod_{j=2}^{m}(p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]}$$

$$= \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[p_{i1}^{k} + \sum_{t=m}^{n} p_{it}^{k}\right]}{\left[\prod_{j=2}^{m-1} (p_{i1}^{k} + \sum_{t=j}^{n} p_{it}^{k})\right]\left[p_{i1}^{k} + \sum_{t=m}^{n} p_{it}^{k}\right]\left[\sum_{t=m}^{n} p_{it}^{k}\right]}$$

$$= \phi^{+}_{m-1} \quad \blacksquare$$

LEMMA C.2

$$\mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im}) = \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

PROOF:

$$\mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix}) = \frac{\left[\prod_{j=1}^{n} p_{ij}^{k}\right](p_{ix}^{k})}{\left[\prod_{j=1}^{m+1}\left(\sum_{t=j}^{n} p_{it}^{k}\right)\right]}$$

$$\therefore \sum_{x=m+1}^{n} \mathrm{Prob}_{m+1}(R_{i1}R_{i2}\ldots R_{im}R_{ix})$$

$$= \frac{\left[\prod_{j=1}^{n} p_{ij}^{k}\right]\left[\sum_{x=m+1}^{n} p_{ix}^{k}\right]}{\left[\prod_{j=1}^{m}\left(\sum_{t=j}^{n} p_{it}^{k}\right)\right]\left[\sum_{t=m+1}^{n} p_{it}^{k}\right]}$$

$$= \mathrm{Prob}_m(R_{i1}R_{i2}\ldots R_{im})$$

We shall now proceed to the proof of Theorem C.1.

PROOF OF THEOREM C.1:

Case 1 : to show that prob[π_i] satisfies (C1):

Rearranging (C1) becomes

$$\frac{\sum_{j=2}^{n} p_{ij}^{k}}{p_{i1}^{k}} \mathrm{Prob}[R_{i1}R_{i2}\ldots R_{im}]$$

$$= \sum_{j=2}^{m} \mathrm{Prob}[R_{i2}\ldots R_{ij}R_{i1}R_{i(j+1)}\ldots R_{im}]$$

$$+ \sum_{x=m+1}^{n} \mathrm{Prob}[R_{i2}R_{i3}\ldots R_{im}R_{ix}] \qquad \ldots\ldots\ldots\ldots(\mathrm{C.1a})$$

Substituting prob[π_i] into L.H.S. of (3.3a) gives

$$\frac{\sum_{j=2}^{n} p_{ij}^{k}}{p_{i1}^{k}}\left[\prod_{j=1}^{m}\frac{p_{ij}^{k}}{\sum_{t=j}^{n} p_{it}^{k}}\right] = \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[\sum_{j=1}^{n} p_{ij}^{k}\right]\left[\prod_{j=2}^{m}\left(\sum_{t=j}^{n} p_{it}^{k}\right)\right]}$$

Substituing prob[π_i] into R.H.S. gives

$$= \sum_{j=2}^{m-1} \mathrm{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}] + \mathrm{Prob}[R_{i2}...R_{im}R_{i1}]$$

$$+ \sum_{x=m+1}^{n}\frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left(p_{ix}^{k}\right)}{\left[\prod_{j=2}^{m+1}\left(p_{i1}^{k}+\sum_{t=j}^{n} p_{it}^{k}\right)\right]}$$

$$= \sum_{j=2}^{m-1} \mathrm{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}]$$

$$+ \frac{\left[\prod_{j=2}^{m} p_{ij}^{k}\right]\left[p_{i1}^{k}+\sum_{x=m+1}^{n} p_{ix}^{k}\right]}{\left[\prod_{j=2}^{m}\left(p_{i1}^{k}+\sum_{t=j}^{n} p_{it}^{k}\right)\right]\left[p_{i1}^{k}+\sum_{t=m+1}^{n} p_{it}^{k}\right]}$$

$$= \sum_{j=2}^{m-2} \mathrm{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}]$$

$$+ \mathrm{Prob}[R_{i2}...R_{i(m-1)}R_{i1}R_{im}] + \phi^{+}_{m}$$

$$= \sum_{j=2}^{m-2} \mathrm{Prob}[R_{i2}..R_{ij}R_{i1}R_{i(j+1)}...R_{im}] + \phi^{+}_{m-1}$$

(by applying Lemma C.1)

$= \phi_2$ (by applying Lemma C.1 m-3 times)

$$= \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[p_{i1}^{k} + \sum_{t=2}^{n} p_{it}^{k} \right]\left[\prod_{j=3}^{m} \left(\sum_{t=j}^{n} p_{it}^{k} \right) \right]}$$

$$= \frac{\prod_{j=2}^{m} p_{ij}^{k}}{\left[\sum_{t=1}^{n} p_{it}^{k} \right]\left[\prod_{j=3}^{m} \left(\sum_{t=j}^{n} p_{it}^{k} \right) \right]}$$

$\therefore$ *R.H.S. = L.H.S.*

Case 2 : to show that (C2) is satisfied.

The following follows from Lemma C.1 :

$$\Sigma \, \mathrm{Prob}_m(R_i \ldots) = \mathrm{Prob}_1(R_i), \quad 1 \le m \le n$$

as in Lemma 2.2.

$$\Sigma \, \mathrm{Prob}[\pi \text{ with } R_i \text{ in position } 1] = \mathrm{Prob}_1(R_i) = \frac{p_i^{k}}{\sum_{j=1}^{n} p_j^{k}}$$

$\therefore$ Σ all possible configurations

$$= \sum_{i=1}^{n} \{ \Sigma \, \mathrm{Prob}[\pi \text{ with } R_i \text{ in position } 1] \}$$

$$= \frac{\sum_{i=1}^{n} p_i^{k}}{\sum_{j=1}^{n} p_j^{k}} = 1 \quad \blacksquare$$

Appendix D. Simulations

In Chapters 2 and 3, a good approximation of the retrieval cost for any self-organizing heuristics is obtained by evaluating the probabilities of each state at successive times t, where state probabilities at time t-1 are multiplied by the transition matrix. The long term probability of each state is approximated by the probability at large t. Let $\pi_i = [R_{i1}R_{i2}...R_{im}]$ and prob$[\pi_i]$ denotes the long term probability of state π_i, the retrieval cost is given as follow :

Expected number of buffer lookups, $\mu(m)$,

$$= \sum_i \{ [\sum_{j=1}^{m} jp_{ij}] + [m (1 - \sum_{j=1}^{m} p_{ij})] \} \text{Prob}[\pi_i]$$

$$= \sum_i \{ m + \sum_{j=1}^{m} (j-m) p_{ij} \} \text{Prob}[\pi_i]$$

Expected number of auxiliary memory fetches, $\nu(m)$,

$$= \sum_i [1 - \sum_{j=1}^{m} p_{ij}] \text{Prob}[\pi_i]$$

We can see that the above method yields a very close approximation to the actual retrieval cost, however, the shortcomings include the huge number of computations required in successive multiplication of the transition matrix, and the enormous amount of storage required to store the ${}^nP_m$ states and the transition matrix. The other method of examining the behavior of various heuristics, though not

as accurate, is by simulation. The next two paragraphs describe how the simulation is done.

To study the cost of a self-organizing heuristic on a linear search system with n elements and m buffers, the initial list is filled with m elements, where the m elements selected (between 1 and n) are randomly generated. The search requests are also produced by a random number generator, distributed according to the retrieval probabilities. Every time a search request is generated, the number of comparisons required to locate the requested element is recorded and the list is reorganized according to the required self-organizing rule. The retrieval cost is then given by the average number of comparisons required to satisfy a search request, and the average number of times a search request is not in the buffer.

To generate search requests, we use a multiplicative congruential random number generator with period 2^{32} to generate random numbers in the range from 0 to 2^{31-1}. This number is then divided by 2^{31} to give a real number between 0 and 1. Suppose each record i has a retrieval probability of p_i, and $\Sigma_{i=1}^{n} p_i = 1$. If the random number generated lies in the interval $[\Sigma_{i=1}^{j-1} p_i, \Sigma_{i=1}^{j} p_i)$, record j will be requested.

In our simulations, each run consisted of at most 1,000,000 searches. After every 5000 searches, the average retrieval cost per search request was recorded, and was compared to the cost last recorded at the previous

5000-search mark. If these two costs did not deviate by more than 0.00001, the simulation terminated. Otherwise, simulation continued until the 1,000,000 searches are completed. The results tabulated in Tables D.1, D.2 and D.3 represent average retrieval costs over two separate runs, using different random sequences to generate the search requests. All simulation results obtained deviate from the average retrieval cost recorded in the Tables by less than one percent.

The simulation results obtained will not be too representative of the actual search cost, as the results greatly depend on the random numbers generated. One would normally expect some discrepency between the simulation results and the actual cost. The simulation results generally demonstrate the superiority of the transposition heuristic over the corresponding move to front heuristic. However, the retrieval costs for the simple k heuristic and batched k heuristic are not very distinguishable. Referring back to the costs tabulated in Chapter 3, we find that when the number of elements gets larger, the retrieval costs for both the simple k and the corresponding batched k heuristics are quite similar, especially for the expected number of memory fetches. Therefore, the simulation results do not demonstrate any clear cut superiority between the simple k and the corresponding batched k heuristic. However, the results do demonstrate the superiority of the k in a row heuristics over the corresponding simple heuristics.

Although this simulation study is not too representative of the actual retrieval cost, nevertheless, it does serve as an alternative to study the behavior of the self-organizing heuristics, when it is infeasible to use the enormous amount of computations and storage required to obtain a good approximation of the actual cost.

TABLE D.1 : Simple Heuristics

The simulated retrieval cost, for a list of n elements and m buffers, assuming the retrieval probabilities satisfy Zipf's distribution. The cost is in terms of the μ, the expected number of buffer lookups, and ν, the expected number of memory fetches.

number of elements	buffer size	simulated retrieval cost Move To Front rule	 Transposition rule
5	3	$2.2005\ \mu + 0.2844\ \nu$	$2.0780\ \mu + 0.2581\ \nu$
	5	$2.6126\ \mu$	$2.4315\ \mu$
15	5	$3.7425\ \mu + 0.4488\ \nu$	$3.2600\ \mu + 0.3690\ \nu$
	10	$5.3533\ \mu + 0.1676\ \nu$	$4.4663\ \mu + 0.1375\ \nu$
	15	$5.8210\ \mu$	$4.8335\ \mu$
30	5	$4.0966\ \mu + 0.5989\ \nu$	$3.5597\ \mu + 0.4903\ \nu$
	10	$6.6145\ \mu + 0.3810\ \nu$	$5.4131\ \mu + 0.2978\ \nu$
	15	$8.2336\ \mu + 0.2432\ \nu$	$6.5730\ \mu + 0.1881\ \nu$
	20	$9.2377\ \mu + 0.1414\ \nu$	$7.3121\ \mu + 0.1107\ \nu$
	25	$9.7739\ \mu + 0.0617\ \nu$	$7.7223\ \mu + 0.0502\ \nu$
	30	$9.9395\ \mu$	$7.8640\ \mu$

TABLE D.2 : Simple 2 Heuristics

The simulated retrieval cost, for a list of n elements and m buffers, assuming the retrieval probabilities satisfy Zipf's distribution. The cost is in terms of the μ, the expected number of buffer lookups, and ν, the expected number of memory fetches.

number of elements	buffer size	simulated retrieval cost	
		Move To Front rule	Transposition rule
5	3	$2.0728\ \mu + 0.2409\ \nu$	$1.9873\ \mu + 0.2281\ \nu$
	5	$2.4214\ \mu$	$2.3127\ \mu$
15	5	$3.3548\ \mu + 0.3660\ \nu$	$3.2600\ \mu + 0.3690\ \nu$
	10	$4.6882\ \mu + 0.1417\ \nu$	$4.4663\ \mu + 0.1375\ \nu$
	15	$5.0884\ \mu$	$4.8335\ \mu$
30	5	$3.6632\ \mu + 0.4878\ \nu$	$3.4677\ \mu + 0.4560\ \nu$
	10	$5.7101\ \mu + 0.3125\ \nu$	$5.2840\ \mu + 0.2827\ \nu$
	15	$7.0366\ \mu + 0.2015\ \nu$	$6.4401\ \mu + 0.1805\ \nu$
	20	$7.8644\ \mu + 0.1197\ \nu$	$7.1567\ \mu + 0.1071\ \nu$
	25	$8.3420\ \mu + 0.0541\ \nu$	$7.5663\ \mu + 0.0485\ \nu$
	30	$8.4879\ \mu$	$7.6896\ \mu$

TABLE D.3 : Batched 2 Heuristics

The simulated retrieval cost, for a list of n elements and m buffers, assuming the retrieval probabilities satisfy Zipf's distribution. The cost is in terms of the μ, the expected number of buffer lookups, and ν, the expected number of memory fetches.

number of elements	buffer size	simulated retrieval cost	
		Move To Front rule	Transposition rule
5	3	$2.0513\ \mu + 0.2378\ \nu$	$1.9782\ \mu + 0.2263\ \nu$
	5	$2.3956\ \mu$	$2.3005\ \mu$
15	5	$3.3288\ \mu + 0.3637\ \nu$	$3.1468\ \mu + 0.3392\ \nu$
	10	$4.6511\ \mu + 0.1411\ \nu$	$4.3124\ \mu + 0.1284\ \nu$
	15	$5.0486\ \mu$	$4.6651\ \mu$
30	5	$3.6440\ \mu + 0.4856\ \nu$	$3.4614\ \mu + 0.4562\ \nu$
	10	$5.6821\ \mu + 0.3115\ \nu$	$5.2803\ \mu + 0.2828\ \nu$
	15	$6.9916\ \mu + 0.2004\ \nu$	$6.4358\ \mu + 0.1802\ \nu$
	20	$7.8434\ \mu + 0.1200\ \nu$	$7.1441\ \mu + 0.1062\ \nu$
	25	$8.2947\ \mu + 0.0538\ \nu$	$7.5469\ \mu + 0.0487\ \nu$
	30	$8.4172\ \mu$	$7.6857\ \mu$